COMMODORE USERS

Ahoy!

$2.50/ CAN. $3.00 MAY 1985

48355
K

...PUT YOUR DISK DRIVE IN OVERDRIVE!...

WE RACE THE LEADING
DISK DRIVE ACCELERATORS
• KWIK-LOAD!
• FAST LOAD
• 1541 FLASH
WHICH IS FASTEST?

DOS BOOST! THE
DOS PLUS
ENHANCED WEDGE FOR THE 1541 OPERATING SYSTEM

HELP FOR CLOBBERED FLOPPIES!
DSKDU
DISPLAY AND UPDATE UTILITY FOR THE VIC AND 64

MANIPULATING COLOR MEMORY
IN YOUR ORIGINAL GAME PROGRAMS

ASSEMBLER ESCAPADES
PUTTING ML INTO BASIC

BY THE NUMBERS
BINARY AND HEX SUMMED UP

40 FILES—NO SCROLLING! WITH
TWO-COLUMN DIRECTORY

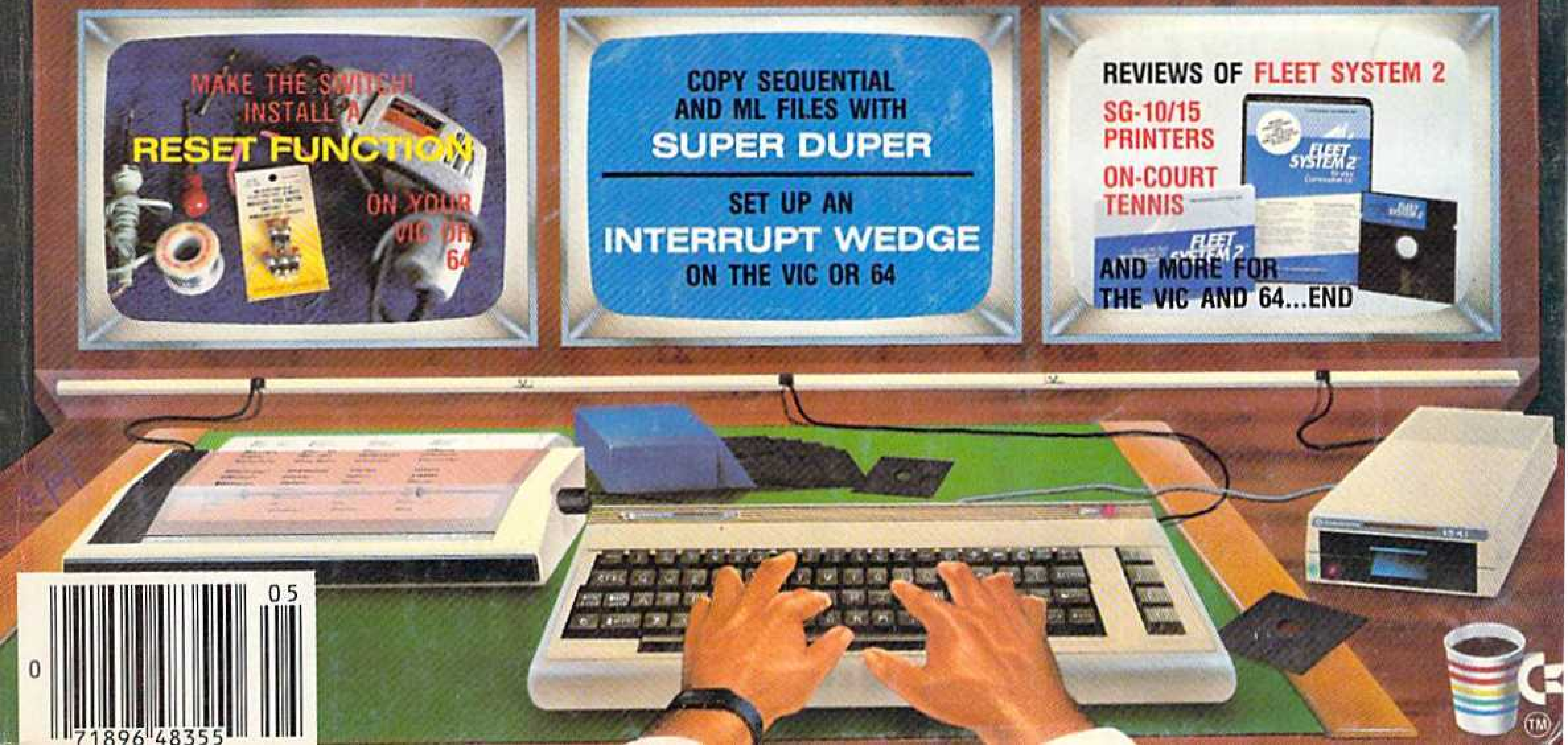
MAKE A
RAID
ON BUTTERFLY-INFESTED HEARTLAND

MAKE THE SWITCH! INSTALL A
RESET FUNCTION
ON YOUR VIC OR 64

COPY SEQUENTIAL AND ML FILES WITH
SUPER DUPER

SET UP AN
INTERRUPT WEDGE
ON THE VIC OR 64

REVIEWS OF FLEET SYSTEM 2
SG-10/15 PRINTERS
ON-COURT TENNIS
AND MORE FOR THE VIC AND 64...END



05
0 71896 48355

Ahoy!

CONTENTS

DEPARTMENTS

FEATURES

*Includes programs: *Worm, Menu* (VIC/64), *Fast Menu* (64)
**Includes program: *By the Numbers* for the 64
***Includes programs: *Listings 1-5* for the 64

PROGRAMS

Cover art by James Regan; cover photography by Morton Kevelson, Liz Benford

Publisher
Michael Schneider

Editor
David Allikas

Managing Editor
Robert J. Sodaro

Senior Editor
Tim Moriarty

Technical Editor
David Barron

Consulting Editors
Ben Bova
Morton Kevelson
Dale Rupert

Art Director
Raoul Tenazas

Creative Director
JoAnn Case

Art Production
Christopher Carter
Mark Kammerer
Eve Griffin

Technical Advisor
Bob Lloret

Circulation Director
W. Charles Squires

Advertising Director
Lynne Dominick

Director of Promotion
Joyce K. Fuchs

Controller
Dan Tunick

Managing Director
Richard Stevens

Advertising Representative
JE Publishers Representative
6855 Santa Monica Blvd., Suite 200
Los Angeles, CA 90038
(213)467-2266
Boston (617)437-7628
Dallas (214)660-2253
New York (212)724-7767
Chicago (312)445-2489
Denver (303)595-4331
San Francisco (415)864-3252

ISSUE NO. 17 MAY 1985

Ahoy! (#8750-4383) is published monthly by Ion International Inc., 45 W. 34th St., Suite 407, New York, NY, 10001. Subscription rate: 12 issues for $19.95, 24 issues for $37.95 (Canada $26.95 and $49.95 respectively). Second class postage paid at New York, NY 10001 and additional mailing offices.

Postmaster, send address changes to *Ahoy!*, 45 W. 34th Street, Suite 407, New York, NY 10001. Direct all address changes or matters concerning your subscription to *Ahoy!*, P.O. Box #341, Mt. Morris, IL 61054. All editorial inquiries and software and hardware to be reviewed should be sent to *Ahoy!*, 45 W. 34th St., Suite 407, New York, NY 10001.

VIEW FROM THE BRIDGE

On the night they hand out awards for speed, versatility, and dependability, the Commodore 1541 need not rent a tux. Third party drives like the MSD SD-2 and the Indus GT have bested it in every category but price. If that last word prompted you to say, "Aha! But see?", then you probably wouldn't even consider upgrading. And why should you? For the price you probably paid, your Commodore 1541 does a more than adequate job. And it will do an even better job after you soup it up with the programs in this issue of *Ahoy!*

• The most important program, unfortunately, you'll have to buy. In the conclusion to *Disk Spinners*, Morton Kevelson examines three best-selling drive accelerators–Fast Load, *Kwik-Load!*, and 1541 Flash–and helps you choose the package that's right for your needs. (Turn to page 33.)

• Once you've sped up your 1541 you can speed up your ability to access it with *Dos Plus*. Dennis Muscatelli's C-64 utility puts a powerful tool kit at your fingertips. (Turn to page 39.)

• Examining and modifying disk files is interesting and useful. Going through standard file processing routines to do so is tedious. Mark Johansen's Disk Display and Update Utility (*DSKDU* to you) gives you direct access to any sector on a disk. (Turn to page 25.)

• You'll never be able to bounce bullets off your disk drive. But *Super Duper* by Dave Barron (not *Ahoy!*'s Technical Editor, but a Mississippi-based programmer) will at least let you copy sequential or program files, rename files, and dupe ML programs without knowing starting and ending addresses. (Turn to page 29.)

• If there's anything worse than a word processor that scrolls horizontally, it's a disk directory that scrolls vertically–namely, any C-64 disk directory that contains 25 or more entries. But *Two-Column Directory* by Richard Lovett and Barry Archer will let you view 40 file names at a glance. (Turn to page 30.)

We figure that's about all the sprucing up your disk operations can handle in one month. As for the rest of the programs and features in this issue:

• The *Commodore Font Editor* by Adventure International programmer Phillip Case will allow you to easily create your own personalized characters or character sets on the C-64. (Turn to page 82.)

• Cal Overhulser's article/program on the *Hardware Interrupt Wedge for the VIC 20 and C-64* will teach you to set up and utilize a "wedge" into either computer's interrupt handler and perform various actions in the background of your main program. (Turn to page 52.)

• Ready for fun and games? So are we! Larry Murdock's *Tile Time* for the C-64 is similar to those sliding number puzzles you used to buy at the dime store–but with infinitely more possibilities and infinitely less wear on your thumbs. (Turn to page 81.)

• Don't get the butterflies playing *Raid* for the C-64, by Larry Coats and Noel Nyman. More important: don't let the butterflies get *you*! They're out to do just that as you try to scoop up all the inhabitants of Heartland. (Turn to page 60.)

• It's been four months since you took soldering iron and drill in hand to accompany Tom Benford, the original hardware hacker, on a computer-carving adventure. This month he shows you *How to Install a Reset Switch on your VIC or 64*, permitting you to clear screen and memory without interrupting power to the rest of your computer. (Turn to page 13.)

• The thought of drilling into the side of his precious C-64 agonized Morton the K to such an extent that he decided to contribute a companion article to Tom's, detailing alternate ways of installing a reset, or "warm boot," capacity. Morton's love for his computer is evident in the very title of *How Many Ways Can I Reset Thee? Let Me Count the Ways.* (Turn to page 16.)

If you don't know assembly/machine language, and if you missed our April issue, we recommend that you borrow a copy and read the first installment of Mark Andrews' *Commodore Roots*. Then read the second installment, found in this issue:*By the Numbers*. There's no more painless introduction to assembly language programming to be found anywhere. Included this month is a program that will easily convert numbers from decimal to hexadecimal to binary. (Turn to page 47.)

And while your brain is in a machine language mode, read this month's *Rupert Report*. Dale takes you on an assortment of *Assembler Escapades*, showing you how to use an assembler to create an ML program, use a disassembler to view that program in memory, and convert the ML instructions into the DATA statements of a BASIC program. (Turn to page 55.)

A passel of programs are likewise included in this month's edition of *Creating Your Own Games on the VIC and 64*, in which Orson Scott Card details the logistics of *Moving Through Color Memory*–and manipulating colors separately from characters in your original game programs. (Turn to page 18.)

Do we envy you readers. As we write these words, it'll be several weeks before we can page through a copy of the May *Ahoy!* You can tear into one right now. What are you waiting for?

–David Allikas

New Number for Ahoy! BBS (as of April 15): 718-383-8909. (See page 77.)

SCUTTLEBUTT

4K RAM ROBOT • GENERAL ELECTRIC PERIPHERALS • REAL ESTATE ADVISOR • ALPHACOM PRINTER • CARTRIDGE COPIER • BOOK RELEASES • NEW GAMES BY ELECTRONIC ARTS, MICROPROSE, GAMESTAR • MUSIC INSTRUCTION • BULLETIN BOARD NEWS FROM ECHO, COMPUSERVE • UNDERWARE RIBBON • RESET BUTTON

CARTRIDGE COPIER

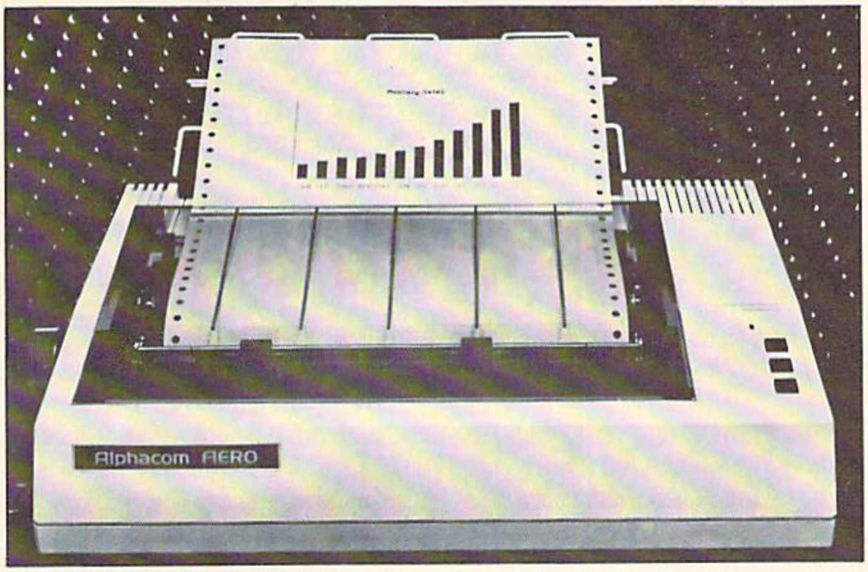
In an upcoming issue, Morton Kevelson will report on various products that let you put your C-64 programs on cartridges. Among these will be the new CBUS cartridge backup system from R.J. Brachman Associates.

The system consists of two parts: CBUS-1, which lets you copy the contents of any C-64 cartridge onto diskette; and the CBUS-2 cartridge emulator, a cartridge containing 16K RAM which allows you to load the previously copied cartridge program back into the computer and completely emulate the original.

While makers of cartridge-based software are sure to denounce CBUS for its inevitable promotion of piracy, the system does in fact provide two important advantages for law-abiding users. First, it eliminates the need to continuously swap cartridges to change programs, saving wear and tear on the expansion port. Second, it provides a valuable development tool for those who want to create cartridge-based software by allowing for rapid testing and emulation of cartridge-based programs without the need to continuously erase and reprogram read-only memory chips (PROMs).

Cost of CBUS 1 is $34.95, $49.95 for deluxe version with accessory programs on disk. CBUS 2 is $84.95, $99.95 including disk. A combination package, including all programs and hardware, is $119.95. Advanced users should inquire about the technical manual for the system.

R.J. Brachman Associates, Inc., P.O. Box 1077, Havertown, PA 19083 (phone: 215-622-5495; for orders 1-800-CBUS-C64).



Alphacom Aero: 130 cps print speed, 2K data buffer, path-seeking logic. READER SERVICE NO. 196

ALPHACOM PRINTER

The $249.95 Alphacom Aero offers 130 character per second print speed, 2K data buffer, and path-seeking logic. 6 X 8 characters are printed with true descenders in a 6 X 9 dot matrix field. The 9-wire print head (using square needles for maximum image quality) prints either 480 or 960 dots per line in alphanumeric (allowing 96 ASCII characters) or bit-image mode. Users may create their own character sets, and mix type styles (double strike, enlarged, etc.) on the same line. Models for both RS232 and Centronics interfaces are available.

Additionally, Alphacom has cut the price of the intelligent printer cartridge interfaces that link its Alphapro 101 daisy wheel printer (see February's *Scuttlebutt*) to various computers. Price of the particular interface required by Commodore users has dropped from $49.95 to $29.95.

Alphacom, Inc., 2323 South Bascom Avenue, Campbell, CA 95008 (phone: 408-559-8000).

WHEN THE RAIN COMES...

Rainy Day Fun is the first in Thorn EMI's new Computer Clubhouse series, designed for children aged 5-9 with the goal of providing activities that entertain while teaching a variety of personal and social skills.

The two-disk set ($39.95) features 10 C-64 programs: *Mask Maker* (choose from 36 different face parts to create over 700 different masks that can be printed and worn), *Paper Airplane* (10 designs to print, with animated forward and backward rolls), *Pin the Tail on the Donkey* (find visible and invisible donkeys via musical clues), *Connect-the-Dots* (create 40 pictures by connecting dots freehand or on the computer), *Party Invitation Maker* (create your own or pick from three predefined



Keep the tykes out of your hair.
READER SERVICE NO. 197

designs), *Sliding Puzzles* (alphabet, number, and picture puzzles solved by moving squares around a 4 X 4 grid), *Banner Maker* (print banners up to 40 feet long), *Fortune Teller* (friendly sorcerer answers questions). *Coloring Book* (color onscreen or on a printout), and *Mouse Maze* (help Mortimer the Mouse escape from the labyrinth).

Thorn EMI Computer Software, 3187C Airway Avenue, Costa Mesa, CA 92626 (phone: 714-751-3778).

ED SOFTWARE CATALOG AND NEWSLETTER

DLM's 1985 catalog features descriptions and sample screens of over 40 educational programs, many for the Commodore 64.

DLM, One DLM Park, Allen, TX 75002 (phone: in TX 800-442-4711; rest of US 800-527-4747).

Educational Activities has announced *Software Source*, a quarterly newsletter to be mailed free to all its customers on a quarterly basis. Included are questions and answers, software and hardware tips, and information on the company's latest products.

Educational Activities, Inc., P.O. Box 392, Freeport, NY 11520 (phone: in NY, HI, and AK 516-223-4666; rest of US 800-645-3739).

GREEN ACRES

Designed to assist anyone thinking of becoming one, *Real Estate Investor* will calculate mortgage payments for conventional, adjustable rate and balloon payments, mortgage amortization of principal and interest, depreciation (based on the 1981 Economic Recovery Act schedule), and projected cash flow and taxable income. For the C-64; $35.00.

Jance Associates, Inc., P.O. Box 234, East Texas, PA 18046 (phone: 215-398-0434).

MUSIC LESSONS

Keyboard Crazy consists of four games that teach children music fundamentals: *Melody Machine* (play music while learning notes on the staff and keyboard), *Sneak-a-Peek* (match notes and note names), *Copy Cat* (reproduce notes and songs), and *Mouse Maze* (move the rodent through obstacles by playing the proper note). Included is a coloring book that works in conjunction with the programs. For the 64; $24.95.

Waveform Corporation, 1912 Bonita Way, Berkeley, CA 94704 (phone: 415-841-9866).

PRINTOUT

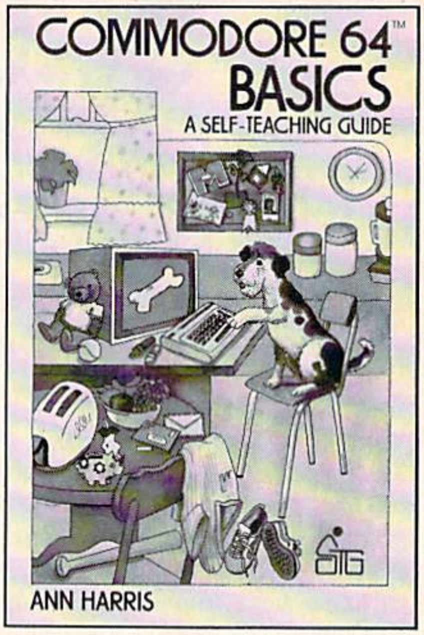
The latest book releases of interest to Commodore users:

Commodore 64 Basics: A Self-Teaching Guide ($15.95) offers a comprehensive course in writing and programming in BASIC, while *Programming Tips for the Commodore 64* ($14.95) teaches the more sophisticated skills needed to create graphics, animation, music, and sound effects. John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158 (phone: 212-850-6000).

Home Applications and Games for the Commodore 64 features 31 programs ranging from home finance to *Beowulf vs. Grendel.* $14.50 from Little, Brown and Company, 34 Beacon St., Boston, MA 02106 (phone: 617-227-0730).

Keyboard Challenge with Commodore 64 ($12.95) combines literacy training with specific subjects in 29 games like *Geography Invaders, Fraction Reaction,* and *Vocabulary War. Assembly Language Programming with the Commodore 64* ($14.95) teaches writing, debugging, and executing assembly language programs. *Commodore 64 BASIC for Kids* ($15.95) provides over 200 exercises for elementary to junior high school students. All from Brady Communications Co., Inc., Bowie, MD 20715 (phone: 301-262-6300).

Bowker's 1985 Complete Sourcebook of Personal Computing lists more than 750 personal computers, 2500 peripherals, 1800 user groups, 545 magazines, 6300 books, and 3300 reviews. $19.95 from R.R. Bowker Company, 205 East 42nd St., New York, NY 10017 (phone: 212-916-1600).



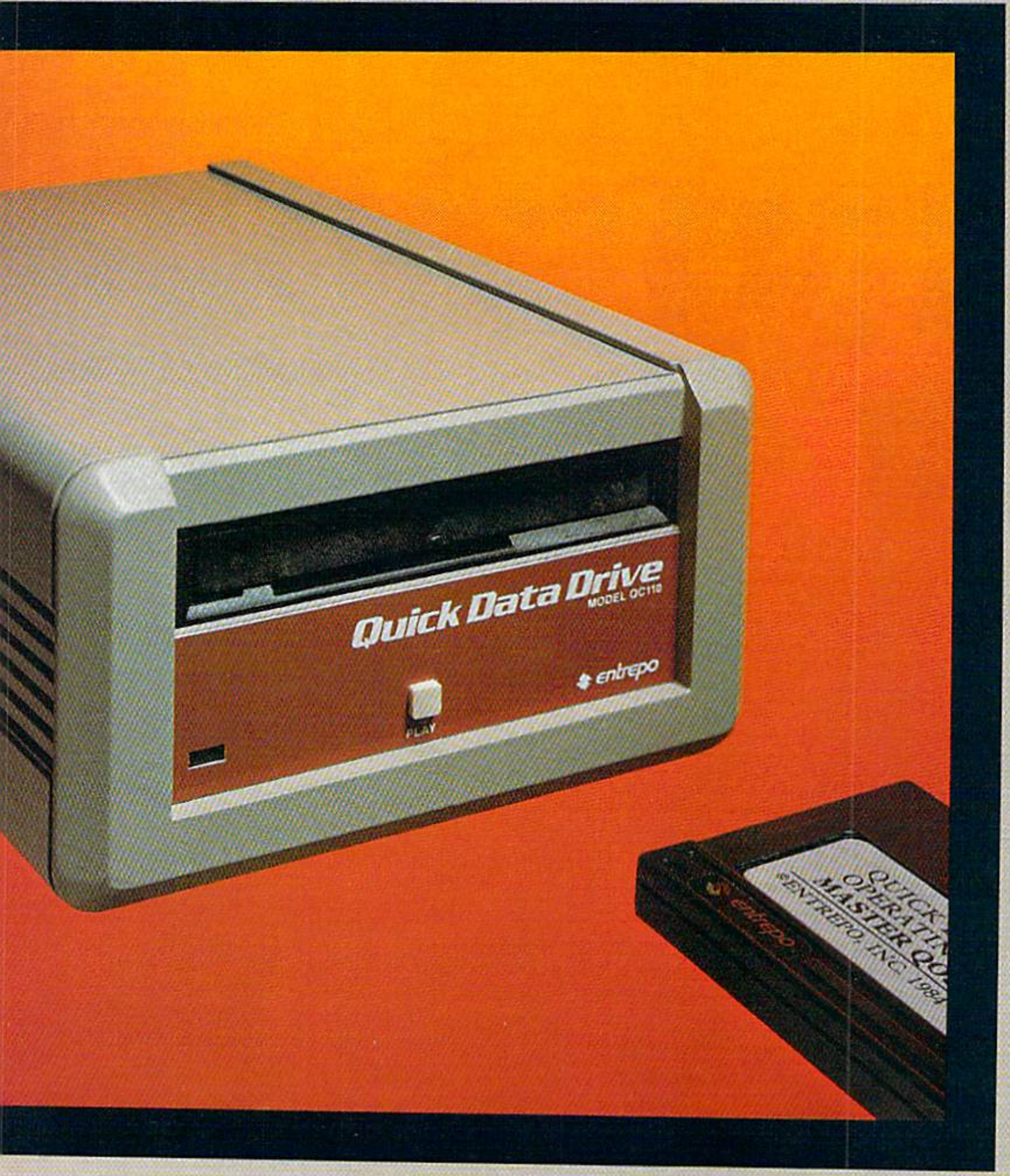
BASIC programming self-taught.
READER SERVICE NO. 198

TELECOM NEWS

The *Sixth Sense* terminal program from Microtechnic Solutions can make decisions and perform actions based on user-programmed parameters (such as time of day and external events), allowing it to do your telecommunicating while you're out. Other features are a 700 line virtual screen with bidirectional scrolling, standard screen for use with built-in editor, split-screen line input, and si-



Wafer Drive. A New Technology



Floppy disk performance for your Commodore® at audio cassette prices.

Entrepo's QUICK DATA DRIVE™ brings high speed mass storage to the Commodore®64™ and VIC 20.™ Designed as a replacement for data cassettes, the QUICK DATA DRIVE plugs into existing cassette ports. You can now load a 24K program in about 20 seconds instead of 8 minutes for audio cassettes and 1 minute for the Commodore Disk Drive.

- Suggested retail price under $85
- 20 times faster than audio cassette
- Fully automatic operation—needs no rewinding
- Highly reliable
- Can be used with a second drive or an audio cassette
- Comes with file management system and utility program
- Top selling software available on Microwafer™ media

entrepo

ENTREPO, INC.
1294 Lawrence Station Road
Sunnyvale, California 94089
(408) 734-3133
Telex 176337

Reader Service No. 213

multaneous downloading to screen, printer, and disk at 1200 baud. On disk for the 64; $89.95 (20% discount for registered owners of *Smart 64 Terminal*).

Microtechnic Solutions, Inc., P.O. Box 2940, New Haven, CT 06515 (phone: 203-389-8383).

The Instant Yellow Pages offers online business information from a database of 6 million listings. Charge is $1.00 per minute and 10¢ per record printed, plus $15 per month subscription fee.

Instant Yellow Page Service, American Business Lists, Inc. 5639 South 86th Circle, P.O. Box 27347, Omaha, NE 68127 (phone: 402-331-7169).

Petscan, a free 24-hour database, provides information on feeding, housing, and health care of animals from pooches to parakeets. Phone number is 303-223-1297.

The Advocate, P.O. Box 8896, Ft. Collins, CO 80525.

CompuServe has replaced their Email service with EasyPlex, offering a 50-name address book for each user, menu, prompt, and command modes, and automatic line numbering.

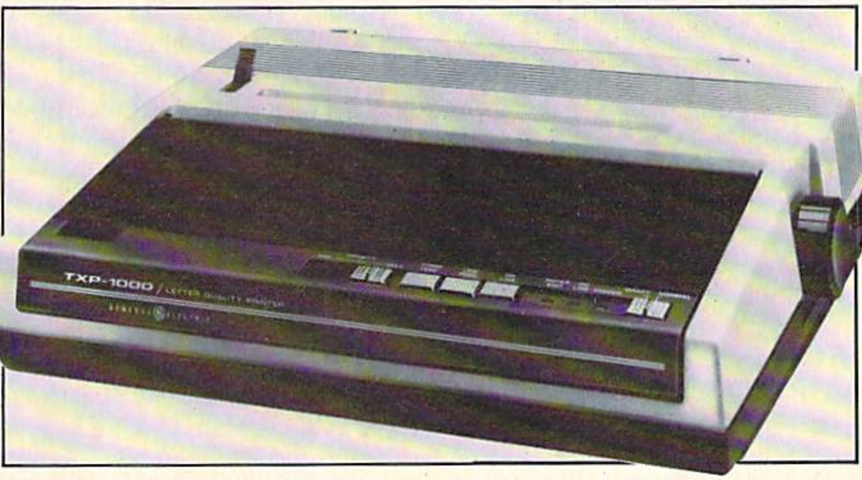
Additionally, CompuServe has made the Academic American Encyclopedia available at a subscription rate of $49.95 per year for unlimited searches (in addition to the normal connect charges).

Finally, Compuserve has announced that beginning in May they will sponsor The Online Computer Connection, a five-minute radio program reaching 100 cross-country markets and allowing listeners to interact with the program via their computers during regular connect times at no additional charge.

CompuServe, 5000 Arlington Centre Blvd., P.O. Box 20212, Columbus, OH 43220 (phone: 614-457-8600).

ECHO Systems has announced Echo Canada, a Toronto-based branch that will spare Canadian users the cost of calling the US. A special rate is being offered to the first 1000 subscribers.

ECHO Systems, 4739 Alla Road, Marina del Rey, CA 90292 (phone: 213-823-8415).



GE's TXP-1000 prints in 16 dot matrix on rolled, fanfold, or cut sheet paper.
READER SERVICE NO. 199

BETTER CONNECTION

The price of The Connection interface from Tymac (see *Printer Interfacing*, March '85 *Ahoy!*) has been lowered from $119.95 to $99.95. It's also been enhanced to configure to most parallel printers via software, and to include a programmable character set which can be downloaded to create various type styles, as well as a universal screen dump for black and white and popular color printers.

Tymac Controls Corporation, 127 Main Street, Franklin, NJ 07416 (phone: 201-827-4050).

GE PERIPHERALS

General Electric's announcement that they will produce peripherals for specific home computer systems (including Commodore) puts the competition among third party manufacturers in a new light. Because GE is the first entrant in the Commodore market capable of swallowing all the other third party manufacturers put together, they have the wherewithal to drive prices down below other companies' profit margins. The prices of the releases described here may not appear trendsetting—but wait until the other giants of the home electronics field follow GE's lead into the Commodore market. Small, independent manufacturers of quality peripherals will continue to sell to the quality conscious; the me-too manufacturers will cease to be.



GE 3-8200 modem is acoustic/modular.
READER SERVICE NO. 200

The model 3-8200 modem, operating at 300 baud, includes both an acoustic coupler for hard-wired phones and a direct mode for use with modular jacks. Other features include automatically switchable answer/originate modes and battery operation. Price: $119.95.

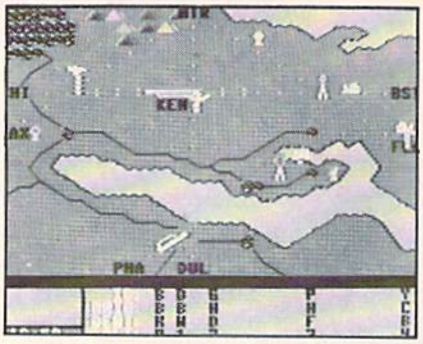
The TXP-1000 printer operates at 25 or 50 characters per second, with full graphics capability and variable width print modes. The $299.95 unit hooks up to the C-64 via an $89.95 interface.

Why no Commodore-compatible disk drive? GE is waiting to perfect the one they're currently working on—which will read data files from all home computers.

General Electric Company, Electronics Park, Syracuse, NY 13221 (phone: 315-456-2446).

NEW GAME RELEASES

Despite the witty title variation, Electronic Arts' *Racing Destruction Set* is one of the two latest entries in



Kennedy Approach: *rule the skies.*
READER SERVICE NO. 205

its Construction Set line, enabling the user to customize and combine a host of supplied elements into an original program. Provided are some of the world's most famous tracks (Monaco, Indianapolis, etc.), nine types of vehicles ranging from Lunar Excursion Modules to stock cars, four different backgrounds, and variable sound effects.

The track can be changed segment by segment, with three widths, seven elevations, and three surface options (ice, dirt, and paving) available. Players can lay land mines and oil slicks in one another's path, or even alter gravity. Vehicles can be changed with regard to speed, tire type, amount of armor, and design. An isolated camera display with split-screen scrolling enables each player to view his car and its position on the track no matter where his opponent is.

Mail Order Monsters lets players design their own beasties from a catalog of 12 body types, 20 enhancements (fangs, tentacles, etc.), 15 weapons, and 6 defenses. Completed monsters can be sent to demolish one another in three different battle scenarios on three skill levels. You can save your most revolting creations to disk for future customization and gameplay.

Both for the C-64; $35.00 each.

Electronic Arts, 2755 Campus Drive, San Mateo, CA 94403 (phone: 415-571-7171).

By putting the entire track onscreen, Gamestar's *On-Track Racing* eliminates the need to pause or restart when one player gets too far ahead of the other (as in scrolling race games) or sacrifice drama by splitting the screen. The competition takes place on 10 famous raceways. For the 64 and 128; $29.95.

Gamestar, Inc., 1302 State Street, Santa Barbara, CA 93101 (phone: 805-963-3487).

The winning entry in Progressive Peripherals' contest challenging users to build levels from the *Wizard* construction set for a second game, the *Wizard Expansion Set* is a more demanding version than the original (required in order to play the *Expansion Set*). For the C-64; $29.95.

Progressive Peripherals & Software, 2186 South Holly, Suite #2, Denver, CO 80222 (phone: 303-759-5713).

Ousted air traffic controllers can wax nostalgic playing *Kennedy Approach*, simulating the control tower situation at various American airports. Up to twenty aircraft can be radioing you at one time (utilizing speech synthesis); you must guide them all to a safe landing. For the C-64; $34.95.

MicroProse Software, 120 Lakefront Drive, Hunt Valley, MD 21030 (phone: 301-667-1151).

Two games previously announced by Tymac, now available for the first time, at $39.95 each:

As *Gandalf the Sorcerer*, you use power bolts to defend your castle from the lizardmen who seek to steal your treasure and drag your apprentices off to Lizard Lair. On disk or cassette for the 64, on cartridge or cassette for the VIC.

Flyer Fox utilizes 3D graphics and software-generated speech to put you in the cockpit of a jet fighter, defending commercial airliners from murderous Migs. On disk or cassette for the C-64.

Tymac Controls Corporation, Franklin, NJ 07416 (phone: 201-827-4050).

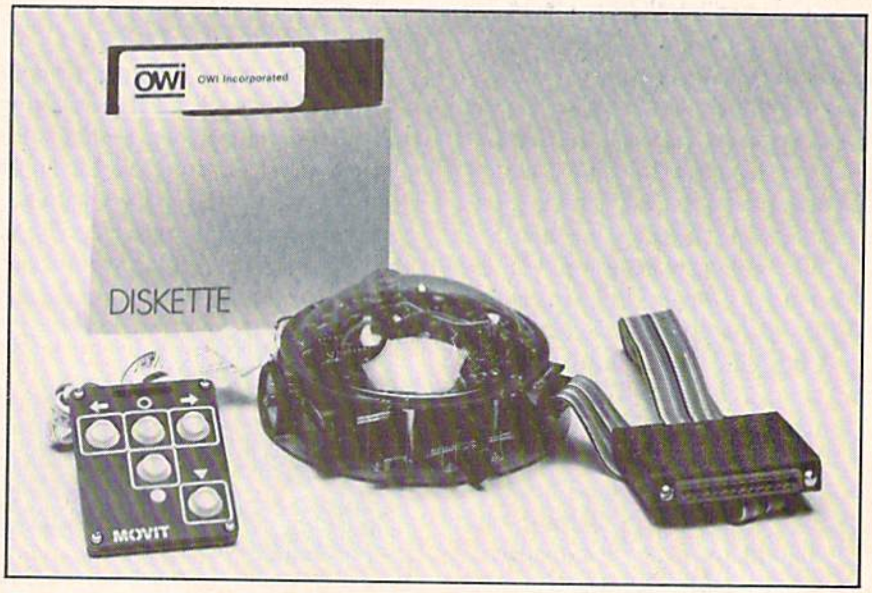
4K RAM ROBOT

Sure to be a riot at the school science fair, The Memocon Crawler is a 5½" diameter robot that can be maneuvered with a seven-function controller or a C-64 (via a special interface). The 4K RAM automaton can go forward, right, or left, pause, sound a buzzer, light an LED, or repeat a program continuously. Robot and controller is $74.95; C-64 interface (assembly required) is $39.95.

OWI Inc., 1160 Mahalo Place, Compton, CA 90220 (phone: 213-638-4732).

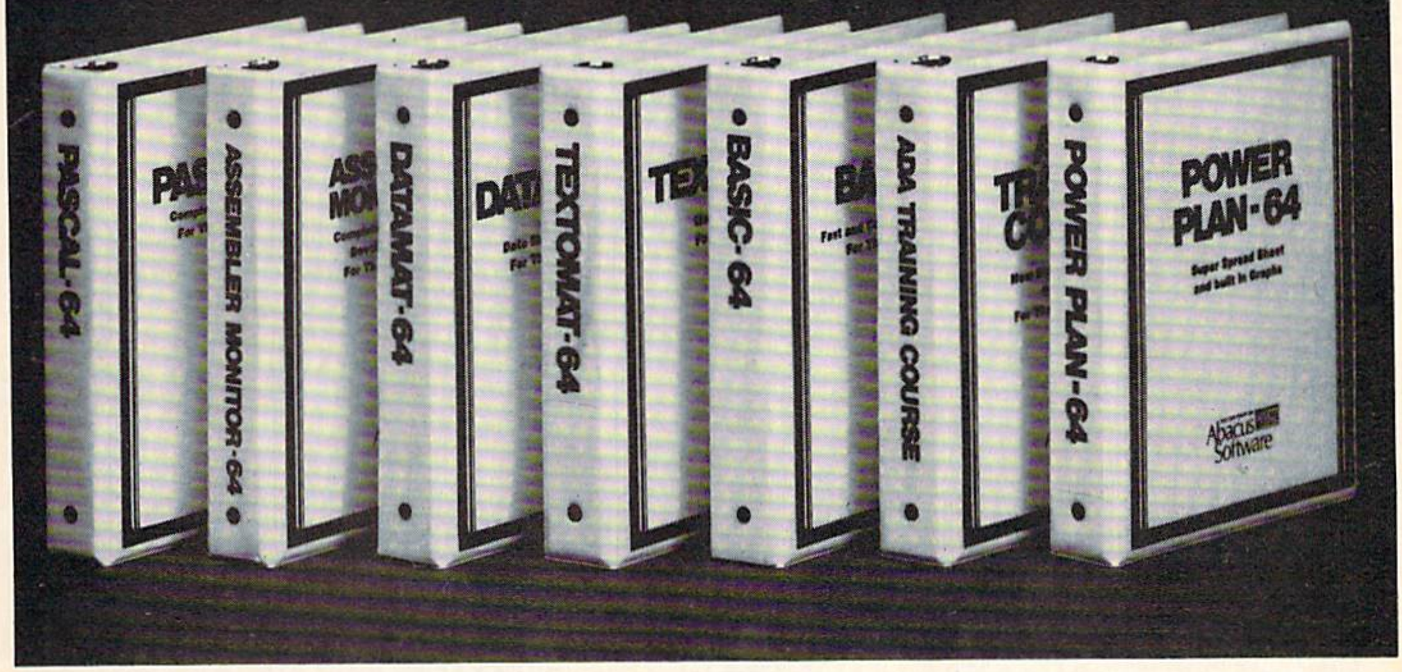
T-64

A column as densely packed with hard news as *Scuttlebutt* needs some diversion. It's provided this month by Diversions Inc., makers of the Underware Ribbon, which allows you



Two printed circuit boards contain all electronics of Memocon Crawler.
READER SERVICE NO. 206

Break the BASIC language barrier



VIDEO BASIC-64 - ADD 50+ graphic and sound commands to your programs with this super development package. You can distribute free RUN-TIME version without paying royalties!
ISBN# 0-916439-26-7 $59.95

BASIC COMPILER 64 - compiles the complete BASIC language into either fast 6510 machine language and/or compact speedcode. Get your programs into high gear and protect them by compiling.
ISBN# 0-916439-17-8 $39.95

MASTER-64 - professional development package for serious applications. Indexed file system, full screen management, programmer's aid, BASIC extensions, 100 commands.
ISBN# 0-916439-21-6 $39.95

PASCAL-64 - full Pascal with extensions for graphics, sprites, file management, more. Compiles to 6510 machine code and can link to Assembler/Monitor routines.
ISBN# 0-916439-10-0 $39.95

ADA TRAINING COURSE - teaches you the language of the future. Comprehensive subset of the language, editor, syntax checker/compiler, assembler, disassembler, 120+ page guide.
ISBN# 0-916439-15-1 $59.95

FORTH-64 - loaded with hires graphics, complete synthesizer control, full screen editor, programming tools, assembler.
ISBN 0-916439-32-1 $39.95

C LANGUAGE COMPILER - a full C language compiler. Conforms to the Kernighan & Ritchie standard, but without bit fields. Package includes editor, compiler and linker.
ISBN# 0-916439-28-3 $79.95

ASSEMBLER MONITOR-64 - a macro assembler and extended monitor package. Assembler supports floating point constants. Monitor supports bank switching, quick trace, single step, more.
ISBN# 0-916439-11-9 $39.95

XREF-64 - indispensible tool for BASIC programmer cross-references all references to variable and line numbers.
ISBN# 0-916439-27-5 $17.95

OTHER TITLES ALSO AVAILABLE - WRITE OR CALL FOR A FREE COMPLETE CATALOG
Call today for the name and address of your nearest local dealer.

PHONE: (616) 241-5510

For postage and handling include $4.00 ($8.00 foreign) per order. Money order and checks in U.S. dollars only. Mastercard, VISA and American Express accepted. Michigan residents incl 4% sales tax.

FREE PEEKS & POKES WALL POSTER INCLUDED WITH EVERY SOFTWARE PURCHASE

You Can Count On
Abacus Software

P.O. Box 7211 Grand Rapids, MI 49510 - Telex 709-101 - Phone 616/241-5510

Reader Service No. 214

to produce T-shirt iron-ons on plain computer paper. Software to draw and print the screen image is not included. Available for the Epson, Okidata, and Gemini dot matrix printers, each $19.95 ribbon will yield a minimum of 20 transfers. If you'd like to produce full color iron-ons, a set of five ColorPens for filling in the iron-on images is also available for $19.95.

Diversions Inc., 1550 Winding Way, Belmont, CA 94002 (phone: 415-591-0660).

DO-ALL DISK DRIVE

A $399 dual disk drive soon to be released by World Disk Drive will include an IEEE bus, serial and parallel printer interface ports, and a disk with printer dumps that can reproduce all Commodore characters.

Unlike the MSD SD-2, whose second drive exists primarily for making backups, these twin drives will be independent of each other and able to be addressed separately. The World drive will also boot all Electronic Arts games, which the MSD does not.



Reset Master: a cinch to install.
READER SERVICE NO. 232

An additional $29 will buy a fast-loading cartridge with copy routines, BASIC 4.0, and a DOS wedge.

World Disk Drive, 23501 Ridge Route Drive, Bldg. D, Laguna Hills, CA 92653 (phone: 714-855-1761).

OFF THE HOOK

A postscript to last month's *Ship to Shore*, wherein Cheryl Peterson reported on the plight of Tom Tcimpidis, Los Angeles-based SYSOP facing persecution for telephone fraud over the appearance of an AT&T credit card number on his BBS: the city attorney's office has dropped the case due to lack of sufficient evidence to prove that Mr. Tcimpidis knew the number had been posted.

It was stressed that the decision to drop the case did not reflect any unwillingness to prosecute future SYSOPs whose boards post illegal messages.

RESET SWITCH

If you don't care to try any of the methods suggested in this issue by Tom Benford and Morton Kevelson to *Install a Reset Switch on your VIC or 64* (see page 13), you can take the easy way out and purchase Master Software's new Reset Master.

The switch, which plugs into any serial port on your system, will reset your C-64 or VIC 20 without shutting off the power, and restore con-

Continued on page 83

How to Install a Reset Switch on Your VIC or 64

By Tom Benford
Photos by Liz Benford

They say that people generally have 20-20 hindsight, and this is true even of the design team at Commodore. The most faithful supporters of the Commodore 64 must admit that there are a few areas in which their computer is lacking. One example is the absence of a reset, or "warm boot," switch. On the Plus/4 and C-16 computers, Commodore provides this switch as standard equipment, having learned a thing or two from the C-64 and VIC. What I'm presenting here is an inexpensive, simple way to add on one of these handy gizmos to your present Commodore computer. Since we'll be hooking it up to the user port, the same instructions and connections apply to either the VIC 20 or C-64, which have identical user ports.

What does a reset switch do? It performs almost the same way the power (on-off) switch of your computer does, with a couple of major benefits. One of these is clearing the screen and memory while not interrupting power to the rest of the computer; it allows you to reinitialize the system without having to turn the C-64 off and then back on again. There are many, many uses for such a switch, among which are restarting a game or utility cartridge, or more practical applications such as regaining control of your computer without having to turn it off and then on again. The possible applications of a reset switch for specific programs (both commercial and of your own device) are numerous; in fact, there are literally thousands of uses for this "warm start" (also called "warm boot") capability, and your imagination is the only limit on what you'll use it for.

While this is a perfectly safe and simple modification, it *will* void your warranty, if still in effect. Also note that neither I nor *Ahoy!* can assume any responsibility for damage to your computer by performing this modification on it. If you have any reservations about your soldering sapience, it may be a good idea for you to give this article and your computer to a friend or technician who can complete the project for you.

Construction is fairly simple, with only a basic knowledge of soldering techniques required. The bill of materials is equally minor, and the whole project should take well under one half hour to complete. Let me mention first that the switch we'll be using is a standard Radio Shack item that comes five to a package. You'll only need one switch for this project, so you may want to give the others to friends or members of your user's group. Or, you may want to keep the extras for possible use on future projects we'll be covering: the choice is yours. Now, let's get our shopping and tool lists together!

Parts List (Figure 1):

1 package (5) Miniature Push Button Switches–Momentary SPST Contacts, Normally Open (Radio Shack #275-1547)–$2.49

Approximately 3 feet light single conductor wire (30-gauge AWG wire-wrap wire is ideal)–$.50

Tools Needed (Figure 1):

Medium-heat (30-50 watt) soldering iron
Rosin-core electronics solder
Wire strippers/pliers
Phillips screwdriver
Electric drill with ⅜" bit
Towel



Figure 1



Figure 2



Figure 3



Figure 4

1. *Tools and parts needed.* **2.** *Removing 3 case screws.* **3.** *Disconnected wire harnesses (keyboard and LED).* **4.** *Lifting copper foil tape from shield.*

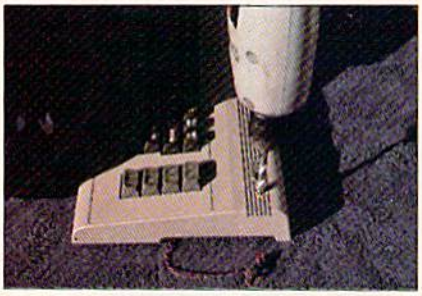


Figure 5



Figure 6

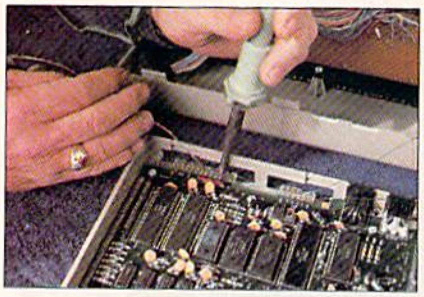


Figure 7



Figure 8

5. *Drilling hole for switch in top cover of C-64.* **6.** *Tinning the wire ends.* **7.** *Soldering wires to the user port—pins #1 (ground) and #3 (reset) are the ones we're using. Solder as close to the inside of the circuit board as possible.* **8.** *Soldering wire ends to switch terminals.*

CONSTRUCTION

Find a clean, well-lit area to work on, spread out the towel, and invert your C-64 on it (the towel will prevent you from marring or scratching the computer). Make sure that *everything* is disconnected from the C-64: power, serial, monitor, cartridge, and user ports should all be empty. Remove the three Phillips-head screws that join the two case halves together, and put these screws somewhere safe for the time being (figure 2). Turn the computer right-side up and *gently* separate the two halves. You'll notice that there are wires connecting the top to the bottom half. These are the LED indicator at the lower right of the circuit board and the keyboard harness at the left side; both of these wires packages use slide-on connectors. Pull gently on the connectors themselves to remove them from the circuit board (figure 3). You should now be able to fully remove the top half of the computer; do this and set it aside for the moment.

Late model C-64s have a paper/foil shield to cut down on TV and radio interference. If yours has this shield, lift the copper foil tape from it and peel it back (figure 4). Once this is done, you'll be able to fold the interference shield forward, thus exposing the heart and soul of the computer—the actual circuit board itself.

To the upper left you'll see the contacts of the user port—we'll be making our solder connections to contacts #1 and #3 (see diagram #1, courtesy of the C-64 owner's manual).

Plug in your soldering iron to get it up to working temperature. While the iron is heating up, we can proceed to drilling the hole for the switch.

Placement of the switch is largely a matter of choice, but I recommend putting it right next to the power indicator LED; it gives a more "professional" appearance, as well as keeping it out of harm's way. Using a ⅜" bit, drill a hole through the top of the computer keyboard about ¾" away from the LED (figure 5). Clean any remnants of drilled plastic from both the outside and inside areas of the keyboard. Remove the nut from the switch and give it a test-fit to make sure it fits in the hole easily (you may have to ream the hole slightly for a comfortable fit).

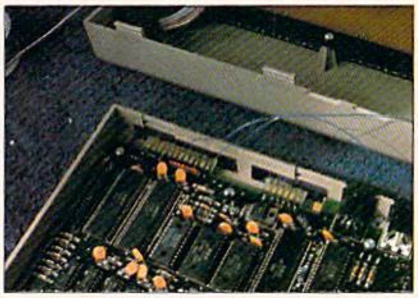


Figure 9



Figure 10



Figure 11

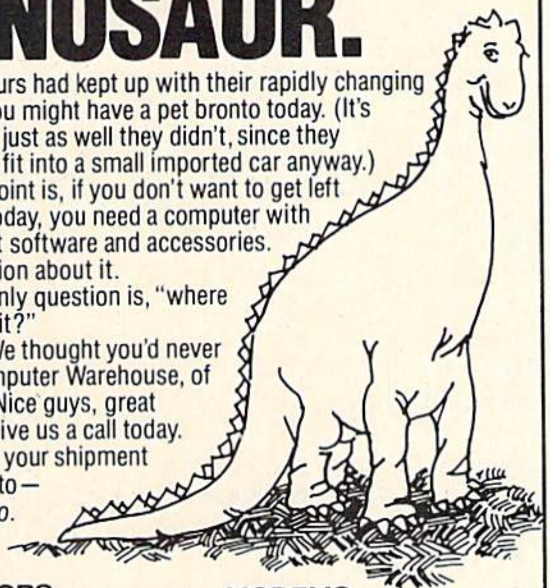


Figure 12

9. *Completed connections to user port pins #1 & #3.* **10.** *Completed connections to switch, with wiring tucked under shield.* **11.** *Just prior to closing: keyboard & LED harnesses back in place, switch mounted on top cover, wiring tucked away.* **12.** *Completed installation. Looks factory-perfect!*

Your soldering iron should be up to working temperature now. Cut your hookup wire into two 16" lengths, and strip about ¼" of insulation from both ends of both pieces. When this is done, "tin" all four wire ends with a small amount of sol-

HOW TO AVOID BECOMING A DINOSAUR.



If dinosaurs had kept up with their rapidly changing world, you might have a pet bronto today. (It's probably just as well they didn't, since they wouldn't fit into a small imported car anyway.)

The point is, if you don't want to get left behind today, you need a computer with the latest software and accessories. No question about it.

The only question is, "where do I buy it?"

Ah! We thought you'd never ask. Computer Warehouse, of course. Nice guys, great prices. Give us a call today. We'll get your shipment out bronto — er, *pronto*.

MONITORS

Commodore 1702	$209.95
Gorilla (green)	$ 79.00
Gorilla (amber)	$ 89.00

PRINTERS

Commodore 1526/802	$209.00
Commodore 803	$169.00
Panasonic 1090	$239.00
Gemini 10X	$249.00
Gemini 15X	$349.00
Power Type (Daisy Wheel)	$369.00
Radix 10	$499.00
ELITE 5 CD (Daisy Wheel) Direct connect for Commodore	$299.00
AXIOM AT 550	$259.00
Star SG10	$269.00
Star SD10	$399.00
Star SR10	$499.00
Okidata 82, 83, 84, 92, 93	CALL
Okimate 10 Color Printer	$199.00
Panasonic 1091	$299.00

DISK DRIVES

Commodore 1541	$199.00
Rana 1000	$199.00
Indus GT	$299.00
MSD Single Drive	$299.00
MSD Dual Drive	$499.00

ACCESSORIES

We carry a full line of covers, ribbons, storage cases, surge suppressors and cables.

MODEMS

Westridge Automodem	$69.95
MPP 1064	$79.95
Total Telecommunication	$69.95
Modem 300 by Commodore	$89.95

PRINTER INTERFACES

Cardco B	$39.95
Cardco G+	$69.95
TYMAC Connection	$79.95
TURBO GT	$69.95
AXIOM Interface	$79.95

SOFTWARE

Flight Simulator II	$36.95
Super Base 64	$59.95
Paper Clip (w/spell)	$79.95
Multiplan by Microsoft	$59.95
Word Pro 3 Plus	$39.95
Koala Touch Tablet	$69.95
Tax Advantage	$69.95
Write Now (Cardco)	$39.95
Mastertronic Disk Game (C64)	$ 7.99

DISKETTES*

Computer Warehouse ss/dd	$14.95
Elephant ss/sd	$13.95
Elephant ss/dd	$16.95
Maxell MD 1	$18.95
Maxell MD2D	$24.95
SKC ss/sd	$13.95
SKC ss/dd	$16.95

*Prices per box of 10

We carry a full line of Commodore Hardware and Software.

COMPUTER WAREHOUSE

(in FL) 305-274-3680 ; 1-800-372-0214

7222 S.W. 117th Avenue, Miami, FL 33183

Add 3% for shipping and handling charges. ($3 minimum). Ad prices reflect 3% cash discount. Credit card orders should add 3%. Prices subject to change without notice.

Reader Service No. 215

der. This is done by melting solder onto the wire ends; not too much, though—just enough to coat the wire ends with solder (figure 6).

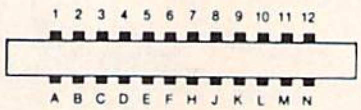
Refer again to the diagram of the user port. With the circuit board facing you, the #1 contact is the one on the extreme right, and #3 is the third one from the right. Solder one end of each wire to each of these contacts, but make sure that you solder them as close to the circuit board as possible. The reason for this is so that the solder/wire won't interfere with the contacts of the port when using a modem or other peripheral. Since you've tinned the wire ends, only a small amount of heat should be necessary to complete the connections (figure 7).

When these connections are completed, take the remaining two wire ends and solder each of them to opposite terminals on the switch (figure 8)—the order doesn't matter on these. The completed connections should look like those in figures 9 and 10.

User I/O

Pin	Type	Note
1	GND	
2	+5V	MAX. 100 mA
3	$\overline{\text{RESET}}$	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	$\overline{\text{PC2}}$	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	

Pin	Type	Note
A	GND	
B	$\overline{\text{FLAG2}}$	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



64 or VIC user port, plus descriptions of pinouts for input/output devices (see **C-64 User's Guide,** ***p. 142).***

Assuming everything's gone smoothly so far, you should now be ready for reassembly. Insert the switch into the hole in the top cover, secure it with the nut, and tighten it up. You won't need the washer, so discard it now.

Place the top half over the bottom half, fold the interference shield back to cover the board, reaffix the copper foil tape to the shield, and plug in the two harness connectors. These connectors are "keyed" so they'll only go on the right way. Don't force them on! When the connectors are positioned over the pins correctly, they should slide right on with very little effort (figure 11). When you've got them in place, close the two halves of the computer.

Now invert the computer, replace the three screws, and tighten them to finish the job. That's it!

The finished job should look like figure 12. The switch looks like it grew there, doesn't it? Plug the power cable back in, as well as the other cables, and you're ready to roll. □

How Many Ways Can I Reset Thee? Let Me Count the Ways.

By Morton Kevelson

I've always been reluctant to perform surgery where it shows on a finished commercial product. This probably stems from an initial conservatism against violating the sanctity of the manufacturer's warranty, or perhaps a misguided urge to preserve the resale value for as long as possible. Whatever the reason, I surmise that many users share this reluctance with me. It thus seems fitting to present a number of ways by which you can add a reset button to the Commodore 64 or the VIC 20 without opening the computer.

If you carefully examine the pinout drawings on pages 142 and 143 of the manual supplied with the Commodore 64 (pages 150 to 152 for the VIC 20) you will notice that the reset line is available at the user port, the serial port, and the cartridge port. In fact, the corresponding pins of each of these ports are electrically identical due to a direct connection inside the computer. A reset switch may be connected between ground and the proper pin at any one of these ports.

THE USER PORT

In the user port, as we have pointed out, the reset line is on pin three. If you simply connect a pushbutton switch between this and ground on pin one the job will be complete. Figure 1 illustrates how to use a 24 pin edge card connector with the pins on .156 inch centers. Properly done, the pin spacing allows the switch to be self mounting. Total cost is about $2.50 for switch and connector. You may want to cover the exposed pins with some insulating tape to prevent electrical shorts due to contact with foreign objects. The proper pins can be easily located by simply orienting the connector as shown in the manual. The pin identification should be molded in the

connector housing, making the job even simpler. Remember, all of the pinouts in the Commodore manuals are as viewed from looking at the back of the computer.

THE SERIAL PORT

The serial port has the reset line on pin six with ground on pin two. If you do not have a disk drive or printer you can actually mount a switch on a six pin DIN connector and install it at the serial port. Public Domain, Inc. (5025 S. Rangeline Rd., West Milton, OH 45383–phone 513-698-5638) offers a completed assembly for just $5. If you have a disk drive, the second port on the 1541 can be used just as easily. If you also have a printer, there is still a way to add a reset switch to this port, although the method is somewhat crude. It will require some minor surgery of the disk drive connecting cable. Carefully slit the insulating cover and remove some of the copper braid which shields the inner conducters. In the case of the cables supplied by Commodore, the wires you want are colored purple and blue. Other manufacturer's cables may have different color code schemes. Simply connect the terminals of the switch directly across the wires. Be careful to maintain continuity of the existing circuit. Figure 2 shows a partially completed job. To finish things off, just wrap the exposed connections with insulating tape.

THE EXPANSION PORT

The cartridge port offers a unique opportunity for adding a reset button. Just find a suitable cartridge, which you intend to leave in place all the time, and install the switch. The Epyx Fast Load cartridge (reviewed in this issue) is a particularly good choice. If you do a soft reset by entering SYS 64738, the Fast Load cartridge is disconnected. If you hit the reset button, the cartridge is reconnected. Figures 3 and 4 show the installation. Of course, you will have to carve up the cartridge somewhat, but cartridges are much cheaper than computers. Some cartridges (such as Fast Load) are held together by a screw hidden under the label. If you feel around the top it will be easy to locate.

PUSHING THE BUTTON

Interestingly enough, when you push the button everything grinds to a halt. It is when you let go that things take off. All the major chips in the C-64, with the exception of the VIC-II chip, are reinitialized. When the button is released, the microprocessor starts executing the code whose starting address is stored in 65532 and 65533 ($FFFC and $FFFD). (For a more detailed discussion refer to the *Exposé of the VIC & 64 Operating System* in the January '85 *Ahoy!*) Just remember that all serial bus peripherals will also be reset by the reset button. It would be a good idea to remove any disks from the drives before initiating a hardware reset.

ON SEMANTICS

The term "Warm Boot" generally refers to a partial reset of a system without turning the machine off and then on. The term "Cold Boot" refers to the startup procedure the computer follows immediately after power on. The reset button discussed in these articles actually has the effect of making the computer go through the initial power on sequence. Thus, some may argue that the switch be called a "Cold Boot" switch, and the term "Warm Boot" reserved for the SYS mentioned above. Others go so far as to argue that even the SYS command should be referred to as a "Cold Boot." We prefer to avoid the whole issue by simply referring to the reset switch as a "Hardware Reset" and the SYS command as a "Software Reset," leaving all temperature considerations to the budding meteorologists in our audience.

Alert readers may have noticed that the word reset in the Commodore manuals is printed with a line above it. This is simply the logical negation symbol. Purists may read this as "not reset." It means that the reset action takes place when this line is a logical 0, or in our case by pulling the five volts normally on this line to ground or 0 volts.

As with all construction projects presented in *Ahoy!*, the user assumes all responsibility for any catastrophies which may result from self-inflicted tinkering. ☐



Figure 1

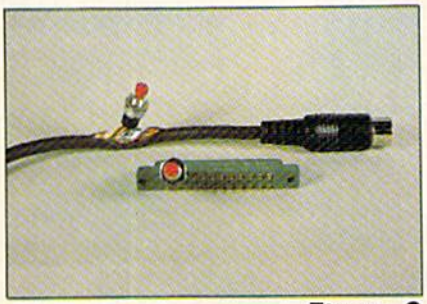


Figure 2

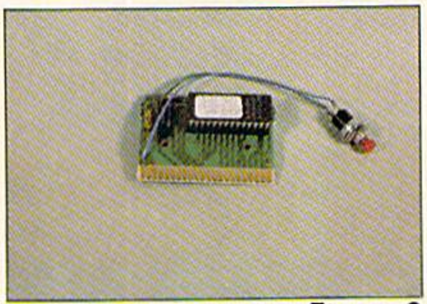


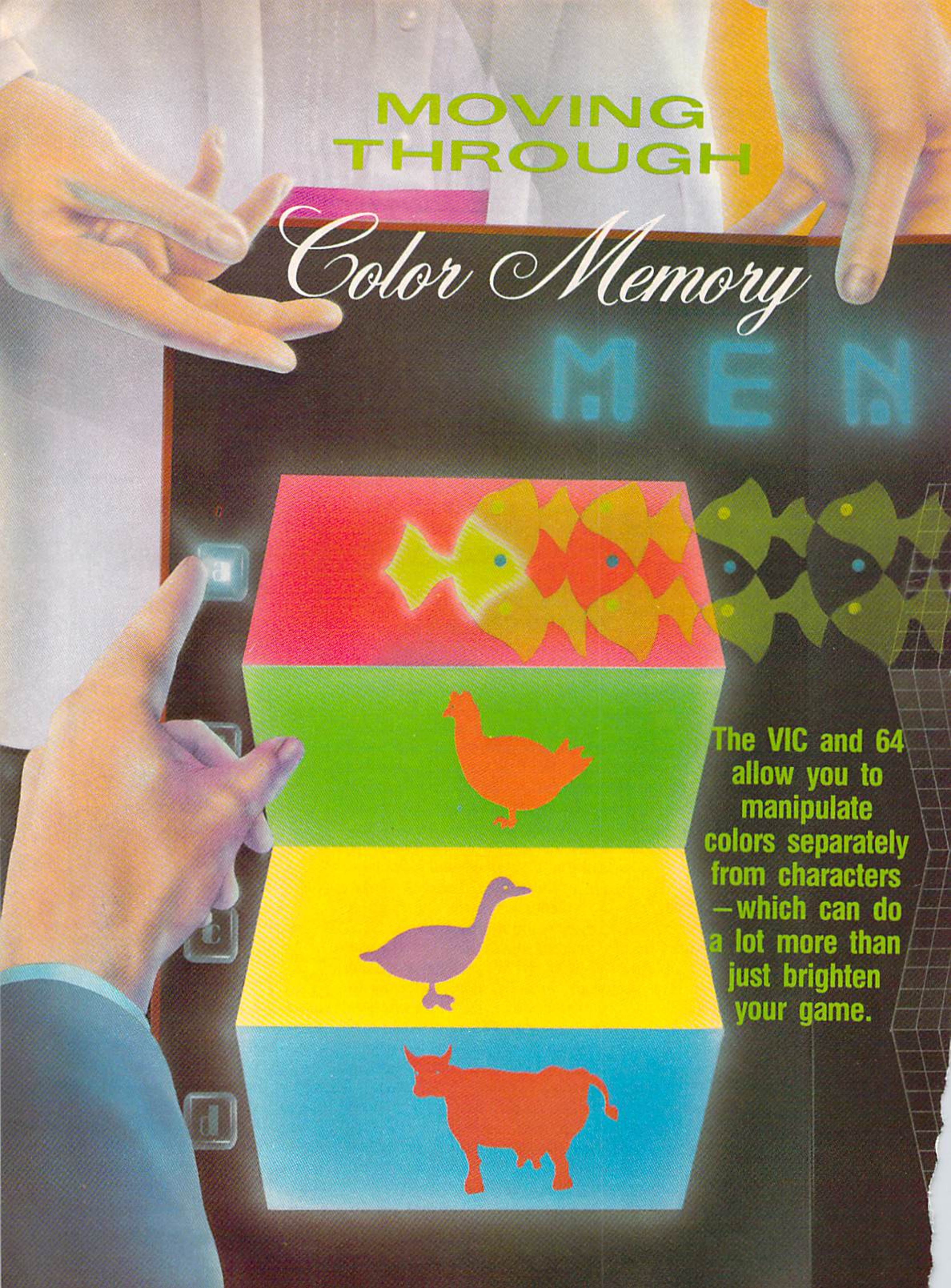
Figure 3



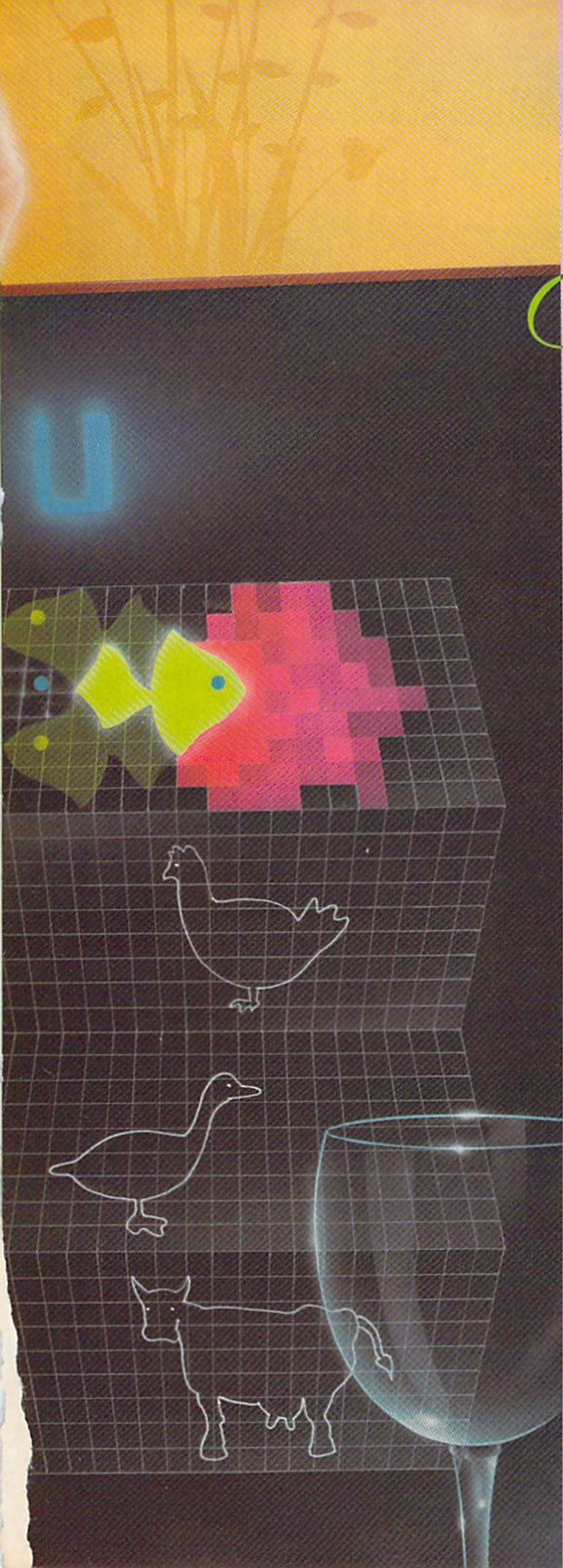
Figure 4

***1.** User port reset: note position of contacts 1 & 3 at left end of top row. **2.** Reset switch installed on serial port cable (top) & user port connector (bottom). **3.** Internal connection of switch in Fast Load cartridge. **4.** Completed assembly.*

MOVING THROUGH *Color Memory*



The VIC and 64 allow you to manipulate colors separately from characters —which can do a lot more than just brighten your game.



James Regan

CREATING YOUR OWN GAMES ON THE VIC AND 64

Recently I was at a party where a lot of science fiction and fantasy writers and readers were gathered. I have no shame—I look at such events as therapy for my ego and I milk them for everything they're worth.

I was told that one of the guests was Walter Meyers, a professor at North Carolina State who happens to be on the panel that gives out the Campbell Memorial Award for the best science fiction novel of the year.

Now, we writers, sophisticated and worldly as we are, all pretend that awards mean nothing to us. But all the time we're saying that, we're swallowing hard to keep from drooling at the thought of getting one.

So I was prepared to discuss, modestly, the significance of my contribution to science fiction and fantasy literature.

Instead, he handed me a disk. "I've been reading your column in *Ahoy!*," he said. "I wondered if you'd look at these games." He mentioned that he had created some really dazzling sound and graphics. And when I got the disk home, darned if he hadn't.

The disk contained several games, but they were all linked with a menu program. The menu began with some messages in nice large characters. Then we were given a musical interlude in Digit's Music Hall. A curtain opened to reveal a beautiful pipe organ. A penguin walked on, sat up to the organ, and played some Bach. The penguin's fingers moved properly on the keyboard; the music was flawless. The curtain went down, and then came back up to reveal the message: "And now, the menu."

Whereupon the letter M walked out to the middle of the screen and beckoned to the other letters. E slithered on, N was a bit more dainty, and U positively wafted. Then, underneath it, five games were offered:

A find-the-letter-on-the-keyboard game, in which nestling birds squawk until their letter is struck.

An abstract GO-like board game, in which players take turns moving tiles and flipping colors.

A duel with swords, in which you select the moves to make against opponents with varying levels of skill.

A memory building (or, in my case, ego-destroying) game in which you try to remember how many Moes, Larrys, and Curlys you saw while a single match was lit.

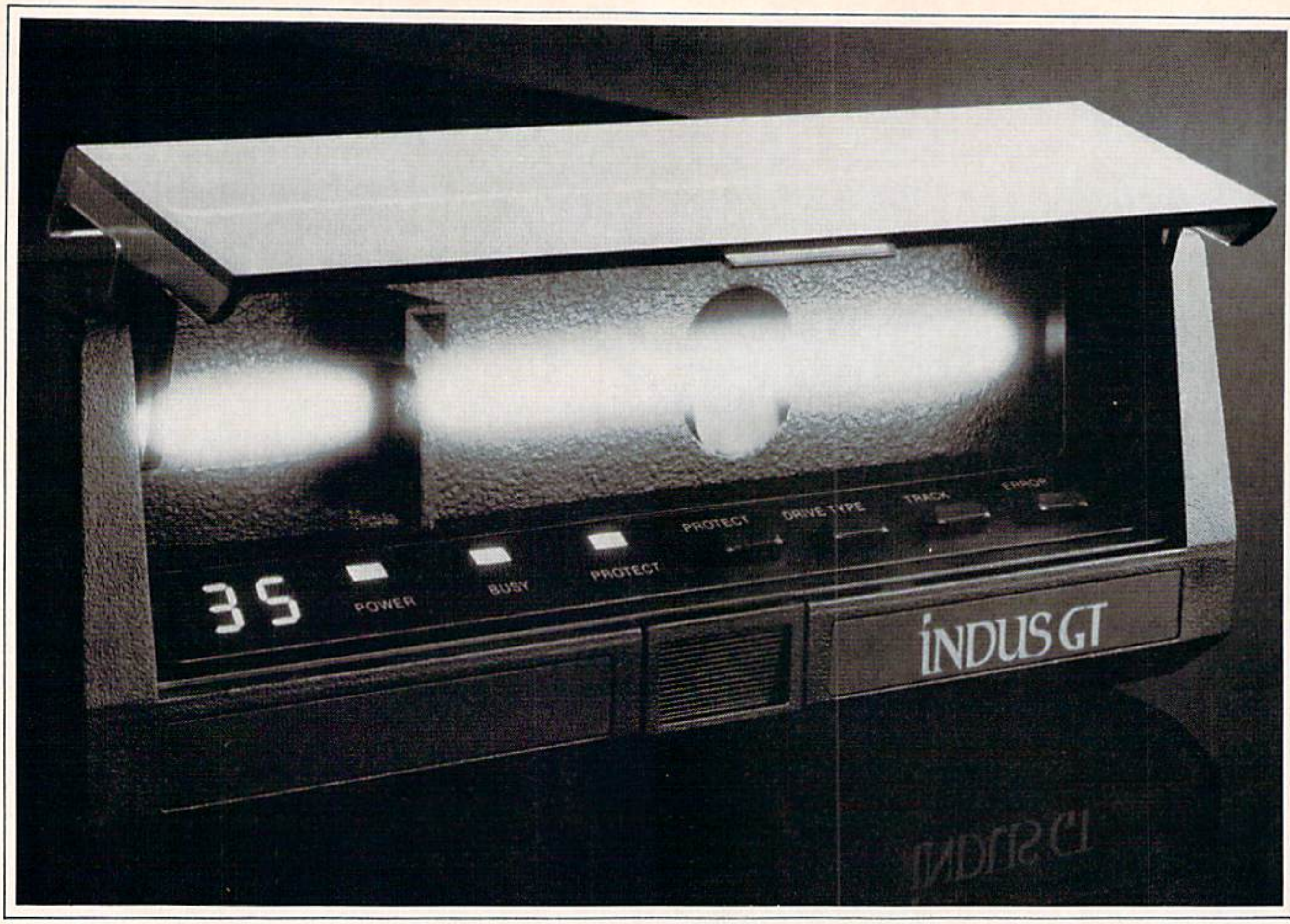
And the *piece de resistance*, a truly magnificent flags-of-the-nations game.

And my job was to play them like an idiot.

GET HELP FROM AN IDIOT SAVANT

You see, once you've programmed your game and it's doing everything you want it to, it's time to have it tested

by Orson Scott Card



What you get if you cross a Commodore 64 with a Ferrari.

You get the incredible Indus GT™ disk drive.

You get brains. You get beauty.

But, that's not all you get.

You get a disk drive that can handle 100% of Commodore's software. Up to 400% faster.

You get the disk drive with the best service record around. With a one year warranty on parts and labor to prove it.

And, you get the only disk drive that comes with free software. Word processing. Spreadsheet. Database manager. Plus, a carrying case that doubles as an 80 disk storage file.

Most of all, you get luxury. From the sleek lines of its soundproofed chassis to the responsive AccuTouch™ controls at the Indus CommandPost™ From the LED display that keeps you in control of your Commodore to the air-piston operated dust cover that protects your disks and drive.

So, you know what you really get if you cross a Commodore 64 with a Ferrari? You get the best.



INDUS™

Reader Service No. 218

by somebody else. Somebody who doesn't know anything about the game. Preferably, somebody who will quickly do everything that you didn't think anybody would be dumb enough to try.

We're talking about genuine creative stupidity.

For instance, in the middle of *Got a Match*, I decided I had better go back to the preschool game, which was better suited to my mental abilities. I tried to quit prematurely. Apparently Walter Meyers hadn't counted on anybody giving up in frustration. There was no orderly exit–at least, when I pressed Q the game didn't Quit, and none of the function keys stopped it, either. Nor did the RESTORE key–but it *did* wreck the display.

What kind of idiot would press the RESTORE key during the game?

Now, Walter Meyers didn't make a mistake, *I* did. But if I had plunked down forty bucks for the game, I wouldn't be rational about something like this. I wouldn't say, "Card, you idiot, why did you press the RESTORE key?" I'd say, "What kind of game is this that the only way you can quit early is to turn off the machine?"

And I'm not only stupid, I'm lazy. I didn't want to have to find the numbers 1 through 5 on the keyboard to choose my guess for the nation whose flag was displayed. I wanted to be able to move a cursor up and down and have the computer highlight the choice I'd made. Then I'd never have to look down at the keyboard at all.

There were times I wanted to do things the game didn't allow. Times I didn't want to do things the game required. I am reporting all those things to Walter Meyers.

Will he act on them all? Maybe. And maybe he'll look at me with pity and wonder how a guy this dumb ever got to write a column for a national computer magazine.

Hear me, O game designers. Treasure your stupid friends. If you have no stupid friends, find some. Because the idiots who do everything wrong when they're testing your game are the biggest help you'll ever get.

By the time you're through making it so they can do all the things they wanted to–and can't do any of the things they wish they hadn't–you'll have a game that is perfectly polished.

You could never make that many mistakes on your own. You just can't do it without the help of us dolts.

SELECTING OPTIONS

Most games use Option Selection at some time during the game. It can be as simple as the One-Player/Two-Player option at the beginning, which is never used again until the game is over; or it can be the heart of the game.

For instance, Walter Meyers' game *Flags* depended entirely on option selection. Each time a new flag was displayed, the names of five nations appeared under the flag, numbered 1 to 5. To select one, you just press a number.

Things got more complex with *Flashing Swords*. Here, there were many options, like Thrust, Parry, Rest, Feint, and so forth. Sometimes a modified set of options appeared, adding Lunge and leaving off some of the others. Now to choose an option you press the appropriate key: L for Lunge, T for Thrust, and so on.

The same thing cropped up in an early version of a trivia game sent to me by Jan Iverson. Possible answers were numbered.

Now, this single-keypress method of option selection is a vast improvement over using INPUT statements to read whole words. For instance, an early game for the Atari was *States and Capitols*. Instead of offering a menu, you had to type in the name of the state. Now, for a touch typist who is a fanatic about correct spelling–like me–that posed no problem. But when my then-four-year-old son played the game, he *hated* typing in all those letters.

What if he spelled Tennessee with only one N? He knew the name of the state–he just didn't know how to spell it. So the game wasn't really a geography teacher–it was a spelling exam.

Worse yet, the programmer tried to "simplify" things by using only the first five letters of the state's or capitol's name in the recognition loop. This meant that the word Washiplvxzyut was as "correct" as Washington. It also meant that if you wrote North Carolina when the state was North Dakota, the program thought you were correct. Great educational program!

So the multiple choice technique, where you are shown several options and allowed to choose by pressing a single key, is a great improvement over requiring the player to type whole words.

But it still has drawbacks. In numbered menus, where you choose option 4 by pressing the 4 key, there is no necessary relationship between the answer and what you press. If the answer you want is number 4, you have to look on the screen to see what number it is, then look down to the keyboard to find the number, and then look back to the screen to see how you did. And don't kid yourself that touch typists don't have to do this. There are an awful lot of us hundred-word-per-minute typists who still stop and look for numbers and symbols on the top row. That's terra incognita up there.

The first-letter option solves part of the problem–T for Thrust means you only have to think of the word Thrust and you know what to press. But it makes things even worse for hunt-and-peck typists. We touch typists know without thinking where T is on the keyboard–but a lot of your players won't know.

There *is* an easier way.

Just put your menu of choices on the screen. Not numbered, but each in its own position on the screen. All the options are printed in letters of the same color–except one. That one is in a much brighter color. It is the current selection.

Then you merely move the joystick up or down. The former current selection returns to the regular color, and the one just above or below it is changed to the highlight color.

You can keep moving from choice to choice, until you

reach the one you want. Then you press the button. You have made your choice. (It works the same way with the keyboard. You use the f5 and f7 keys to move up and down, and the space bar to select.)

Why is this better? Because you want to have as few barriers as possible between the data and the decision and between the decision and the action. In this case, players don't have to figure out which number or letter represents their choice. Their choice is highlighted on the screen. And to change from one choice to another, they don't have to find a particular key on the keyboard. They only have to remember *up*, *down*, and *action*.

These games were good games even with the relatively clumsy control system. But that's what you do when your game is finally working – you fine tune it, change it here, change it there, to make it simpler to play, and yet more fun, more fascinating, more satisfying.

HIGHLIGHTING WITH PRINT

Now that I've described how highlighting with a menu of options is the best way to handle multiple choice games, it's only fair that I show you how it's done.

The screen display is simple. You just put the choices in a column. Let's say we're playing "Ambassador," and you have to negotiate with the minister of state of Lotharingia. Your king has instructed you to engage Lotharingia in an alliance against powerful Bavaria. In the game, you are given bits of information to help your preparation – those appear at the bottom of the screen. Your instructions from the king appear at the top of the screen. And in the middle are your options:

Give gifts
Make threats
Offer concessions
Maneuver
Stall
Leave

Each choice leads to a sub-menu, which appears when you have made your choice. For instance, if you choose to Offer Concessions, the sub-menu lets you choose which concession to make:

Trivial concessions
Lower tariffs
Free trade
Loans
Economic aid
Passive military aid
Active defensive aid
Active offensive aid

Once you have selected from the sub-menu, the opponent (or computer) responds, and you prepare your next move. After a certain amount of time, the computer judges how well you accomplished your objective, and the next round begins.

There are several ways to program the highlighting of options. You could keep every option in a string array, and PRINT the highlighted option each time in a different color. For instance, let's say that the six elements in the initial menu are held in the array OP$*(n)*. There are two colors, C1$ and C2$. C1$ is white, and C2$ is light blue – the normal color for letters on the screen.

To put the initial menu on the screen, you would get the cursor to the beginning of the option menu and use:

```
FOR I=0 TO 5:PRINT C2$OP$(I):NEXT
```

Then, to highlight option 0, the first option, you would get the cursor to the right row and use:

```
PRINT C1$OP$(0)
```

Now option 0 appears in white.

When the player moved the joystick down to the next row, you would first get the cursor to the position of the old choice, option 0, and

```
PRINT C2$OP$(0)
```

That sets the first option back to the regular background color.

Then, get the cursor to the position of the new choice, option 1, and

```
PRINT C1$OP$(1)
```

Now the second option, option 1, appears in white.

This system would work just fine, except that for multiple menus you'd have to set up two-element arrays, with one element to choose the menu and another to choose the option.

But there's a simpler (though not always faster) way, based on the way the Commodore 64 and VIC 20 handle foreground color.

HIGHLIGHTING WITH COLOR MEMORY

As you know, the screen is mapped in "screen memory," a portion of memory in which each byte corresponds with one position on the screen. The bytes of memory contain the code number for the particular character to be displayed in that particular position on the screen. (These are not ASCII codes – they are the screen display codes.)

The first address in screen memory contains the code for the character that will appear in the upper left-hand corner of the screen. The second address in screen memory contains the code for the character that will appear just to the right of it. The 22nd address (on the VIC) or 40th address (on the 64) contain the code for the rightmost character on the top line. The next address has the code for the leftmost character on the next line down. And, of course, the last address in screen memory contains the code for the character in the lower right-hand corner of the screen.

When you use PRINT statements, BASIC takes the

Continued on page 79

CLOSE ENCOUNTERS OF THE FANTASY KIND



QUESTRON™ *A role-playing game so spell-binding, it suspends reality and conjures up an enchanted world whose only hope against the forces of evil is you.*

GEMSTONE WARRIOR™ *A lightning-fast action strategy game that will leave you breathless with excitement as it takes you through the labyrinths of the Netherworld on a noble quest.*

Open them up and step through the gateway to your fantasies come true. At your local computer/software or game store today.

QUESTRON™ is available on 48K diskette for APPLE® II with Applesoft ROM, Apple II+, IIe, and IIc ($49.95). On 40K diskette for ATARI® home computers ($49.95). On 64K diskette for COMMODORE 64™ ($39.95).

GEMSTONE WARRIOR™ is available on 48K diskette for APPLE® II with Applesoft ROM, Apple II+, IIe, and IIc ($34.95). Also on 64K diskette for COMMODORE 64™ ($34.95).

SSI™

If there are no convenient stores near you, VISA & M/C holders can order direct by calling 800-227-1617, ext. 335 (toll free). In California, 800-772-3545, ext. 335. Please specify computer format and add $2.00 for shipping and handling.

To order by mail, send your check to: STRATEGIC SIMULATIONS INC, 883 Stierlin Road, Bldg. A-200, Mountain View, CA 94043. (California residents, add 7% sales tax.) All our games carry a "14-day satisfaction or your money back" guarantee.

WRITE FOR A FREE COLOR CATALOG OF ALL OUR GAMES.

Reader Service No. 211

PHOENIX RED
SOFTWARE



ADVENTURE GAME TUTORIALS

Disk I

TEXT ADVENTURES

- ZORK I
- ZORK II
- ZORK III
- ENCHANTER
- SORCEROR
- STARCROSS
- SUSPENDED
- PLANETFALL
- DEADLINE
- WITNESS

Only $9.99!

Disk II

MEGAVENTURES

- ULTIMA II
- ULTIMA III
- ULTIMA IV
- RETURN OF HERACLES
- QUESTRON

Only $9.99!

Disk III

GRAPHIC ADVENTURES

- GOTHMOGS LAIR
- MASK OF THE SUN
- DEATH IN THE CARIBBEAN
- BLADE OF BLACKPOOLE
- CRITICAL MASS
- GRUDS IN SPACE
- ULYSSES AND THE GOLDEN FLEECE
- WIZARD AND THE PRINCESS
- ASYLUM
- THE INSTITUTE

Only $9.99!

Have you ever played an Adventure Game where you just plain got stuck and couldn't get any further? Well, here's the answer — 'Phoenix Red's Adventure Game Tutorials. They can be used as hints or they will tell you exactly what to do to get out of all those sticky situations. These Tutorials are written for **your** computer (A Commodore 64) not someone elses, so they **work.**

We pay postage and handling.
A must-have for all adventures.
Order yours today!

"We put out money where our software is, not our copy protection."

— Phoenix Red

Send check or Money Order to:

Phoenix Red Software
1620 Bay Area Blvd. Suite 703
Houston, Texas 77058

Call 713-280-8113 for more info.

Reader Service No. 217

DSKDU

A Disk Display and Update Utility for the C-64 and VIC 20*

*5K Memory Expansion Required

By Mark Johansen

Not long ago I had some hardware problems with my 1541 disk drive and, before I realized what was causing my difficulties, the directory on a rather valuable floppy was destroyed. I took the drive to the shop and it was duly repaired, but in the meantime a disk was rendered useless. There is probably a corollary to Murphy's Law that states that when such a thing happens, the floppy disk destroyed will always be the only one you own that you neglected to back up, for that is exactly what happened to me. But I knew that most of the data on the floppy should be intact: it was only the directory and perhaps one or two files that were damaged. If only I could repair the directory, the effort that went into writing several good programs could be saved.

The problems involved in such an effort could be divided into two parts: one, I had to have a way to directly modify the directory; and two, I had to know what I should modify it to be.

To solve the first problem I wrote a disk display and update utility, which I call *DSKDU*, and which I shall endeavor to describe here. It allows the user to look at what is on any given sector on the disk and modify it if he so chooses, while bypassing all the standard file processing routines. It requires a C-64 or a VIC 20 with about 5K expansion and, of course, a 1541 disk drive.

The second problem was made much easier than it might have been by the fact that Commodore provides fairly good documentation in the *1541 Disk Drive User's Guide* on how they laid out their disks.

The results of this effort have proved to be useful in situations beyond the original problem. For example, I have occasionally managed to create files with illegal names such as "," which cannot be deleted with the Scratch command, but which can be killed off with *DSKDU*. On the other hand, it makes it rather easy to "un-delete" files that I have accidentally scratched (provided I get to them before the system has reused the disk space). It has allowed me to look at exactly how various types of files are stored, and I have even used it a few times to create data files that would have been inconvenient to produce with BASIC.

FLOPPY FORMAT

If you hope to accomplish anything useful by directly modifying disk data, you must have some idea of what the results should look like. Commodore conveniently provides details of their disk storage layouts in the *1541 Disk Drive User's Guide*, pages 65-68. I see no need to repeat all the details here, but the manual is extremely terse and some explanation may help you to understand what the tables are all about.

The disk is divided into 35 tracks, numbered from 1 on the outside to 35 nearest the center. Each track is divided into sectors, which are numbered from 0 to 20, but not all tracks have the full 21 sectors – the number of sectors per track decreases as one nears the center of the disk. (The table on the top of page 65 in your *User's Guide* gives the actual number of sectors on each track.) Note that for some mysterious reason Commodore chose to number tracks beginning with one but sectors beginning with zero.

The system maintains a directory on each floppy in which it records what files are on this floppy along with certain information about them. The directory always begins certain information about them. The directory always begins on track 18, sector 1. Each file has one entry in this directory. The first entry begins on the third byte of the block and each entry is 30 bytes long with two byte gaps between them.

The first byte of each directory entry contains a code indicating whether this file is sequential, program, relative, or "user". If this byte is "00" then the entry is not currently in use; it is either beyond the largest number of files you have ever had on the floppy or the file has been deleted. In this case any other information in the

entry is irrelevant.

The most obvious piece of data is the file name. This is easy to see because it is stored as normal ASCII text. The primary function of the directory is to relate this name to the actual data on the disk, which it does by also keeping the track and sector numbers of the first block of the file.

In this first block, the first two bytes contain the track and sector numbers of the second block. The remaining 254 bytes contain the beginning of the actual data making up the file. The first two bytes of the second block contain the track and sector numbers of the third block, which contains the track and sector numbers of the fourth block, and so on, forming a kind of chain. The "pointer" in the last block has a track number of zero. (Remember that "real" track numbers begin with one.) Instead of a sector number, the second byte will contain the number of bytes in this block which actually form part of the file. This allows the system to determine exactly where the file ends; without such an indication all files would have to come in exact multiple of 254 bytes.

The directory itself is chained together just like a file. The first two bytes of its first block contain the track and sector numbers of its second block, which points to its third block, and so on. It wouldn't do if we had to look in the directory to find the directory, so instead it simply always begins at track 18 sector 1. By the way, this location was presumably chosen because it is in the exact center of the disk, and as the directory will normally be referenced more than any other data this should minimize movement of the read/write heads and therefore of disk delays.

This system of chaining works fine for sequential (and program) files, but relative files are intended to be accessed randomly; that is, we do not want to have to read through all preceding records in order to find the one record desired. To enable us to do this, Commodore employs what they call "side sectors." The directory entry for a relative file contains a track-sector pointer to the first side sector in addition to the pointer for the first data block. The first two bytes of this first side sector block point to the second side sector, which contains a pointer to the third, and so on in the now-familiar pattern. Each side sector contains pointers to up to 120 data blocks. Thus, when a program asks for a record, the system will first determine which block it is in by comparing the record length to the standard 254 data-bytes per block. If it turned out to be in, say, block 67, the system would then find the 67th pointer in the first side sector block and retrieve that block directly, without having to follow the chain all the way from the first data block. Similarly, if the record turned out to be in block 700, the system would find the 100th pointer in the sixth side sector block. Now, the whole purpose here is to avoid having to follow the pointers to find the desired block, so we really don't want to have to follow pointers to get to the needed side sector. Instead, each side sector contains pointers to all of the other side sectors. The system will always keep the last one used in memory, so even if it doesn't need the same one again at least it will know where the one it needs is to be found. Each file is limited to a maximum of six side sectors, but as each can point to 120 data blocks, this allows for a total of 720 blocks, more than are actually on a disk.

The format of user files is completely determined by the programmer, so they do not necessarily follow any of the above rules. If you wish to use them, you're on your own.

USING *DSKDU*

The operation of the program is intended to be straightforward. One general point should be made before I describe the individual functions. All numbers are displayed in hexadecimal, and all input must also be in hex. While this may seem inconvenient at times, most notably when entering track and sector numbers, the disk data itself is usually most logically presented in hex, and as you often get the next track/sector from the data in the current one, it seemed reasonable and consistent to make all input and output be in hex.

Once it is LOADed and RUN, the program ask for the first track and sector to be displayed. (Remember these numbers must be entered in hex.) The program will read the disk and display the first sixty-four bytes of that block on the screen. At the extreme left of each line will be the offset into the block, followed by the

An open letter to the readers of Ahoy Magazine Vincent Kurek President: The Ennon Corporation

My purpose in writing is to ask you to join me in shaping the future of the new and most unusual field in computer technology today: Artificial Intelligence.

This incredible power and spectacular creative potential are available to you, for your computer right now. However, there is an alarming possibility that such amazing technology which you have every right to, may not be available to you other that through this offer.

This is unfortunate but somewhat understandable due to the way technology is created. You see, only the business oriented corporation can finance research. It therefore is in a position to dictate immediate research goals. These goals are increasing profits through more efficient production. While valid, they are merely creative and do absolutely nothing to foster exploration in new applications. The result: technology is never used to its fullest potential. But what's worst of all is that these competitve corporations have absolutely no desire to share technology with each other, let alone with you. So, they don't. As a result, the infinitesimal amount of technology that finally trickles down to you is:

A. So expensive you are prohibited from procurring it
B. Shamefully inferior to the real thing

remember...you can buy high-tech consumer goods, but never the technology that creates it.

This same situation confronts you in the new Artificial Intelligence field, but with a difference: There is no true Artificial Intelligence for the home computer user! The few programs claiming to be Artificial Intelligence are really simulators. They are not the real thing. Possessing a mere token of the power and versatility, simulators are clearly not worth their expensive price.

I have tried repeatedly to convince my colleagues that it is in their best interest to release genuine Artificial Intelligence to the general public. The refinement, modification and adaptation as individuals create new applications would improve Artificial Intelligence tremendously. This would benefit everyone in the long run.

I have met with little success. Apparently, it seems that immediate corporate profit is more important than sharing technology with the public. Therefore, the Ennon Corporation stands alone in offering superior Artificial Intelligence programming directly to the home computer enthusiast.

Announcing AN-83: The "Thinking" Program

Believe me when I say AN-83 is the real thing. It is a true "thinking" program that receives an initial "knowledge base" from a data file read when AN-83 is started. Using inductive and deductive logical analysis, this amazing program deduces everything from that data and adds it to its memory. Conversing with you, AN-83 adds and combines with facts already known. It generates new conclusions not explicitly contained in its original knowledge base—just like your own thinking process! The result: it knows considerably more than the specific facts given to it.

AN-83 can also think about anything. It is virtually unlimited in its application. Think of your possibilities. The potential is limitless. In the right hands, AN-83 would revolutionize the adventure, strategy and other smart game-playing programs to say nothing of classic arcade games. On the other hand, AN-83 could be one of the most powerful business analysts available to the home computer.

FREE SOFTWARE

In addition, you will be receiving free, Eliza—the most amazing conversational A.I. program to date. Run this for your friends and jaws will drop with amazement. Eliza's responses are so human, it's uncanny. An entertaining program, Eliza will answer once and for all the question: What can your computer do?

How to Learn Artificial Intelligence

You can be creative. Experiment and modify to fit your personal use because AN-83 and Eliza both possess source code in basic, the most popular easy to use language for the micro. Their extensive, easy to understand commands walk you through the source code step by step. It's surprisingly simple. Even the beginner can understand the "How and Why of A.I."

A Fantastic Savings

The real profit to Ennon Corporation will be your participation in the future of Artificial Intelligence. Therefore, I am pleased to say nobody will miss this chance because they could not afford it. AN-83 is priced to cover just a fraction of its research and developmental costs.

The "Thinking" Program AN-83 is just $21.57. What's more, the astounding Eliza is yours absolutely free.

I guess it's obvious that I want you to participate in the future of Artificial Intelligence. Forgive my excitement and enthusiasm but I just know you are going to be very happy and impressed that such things could be done with your computer. You just won't believe it. Please take this opportunity now. Simply fill out your coupon below and mail today. Don't miss out. It's such a wonderful future of discovery and excitement that awaits you.

With very best of wishes,

Vincent Kurek

Vincent Kurek

☐ Please send me the "Thinking" Program AN-83 for only $21.57.
In addition, I will receive absolutely FREE Eliza—the most impressive conversational Artificial Intelligence program to date.

Available in disk only.
For use with the Commodore 64.

Name ______________________

Address ______________________

City ______________________

State ________ Zip Code ______

THE ENNON CORPORATION
1817 W. Call St., Suite B-8
Tallahassee, FL 32304

Canadian orders must be in U.S. dollars.
We export to other countries.
Commodore is a registered trademark.

values of the data bytes in hex, and then their representations as ASCII characters. Any byte which is not ASCII for a letter, digit, or one of a few special characters will instead be displayed as a period.

Due to the limitations of the VIC's screen size, the program can only display 64 bytes of each 256 byte sector at one time. To get the next 64 bytes you must press the f3 key. If you wish to go back to the preceding 64 bytes you must hit the f1 key. If you hit f3 while the screen is displaying the last 'page' of the block you will be shown the first page, and if you hit f1 while the screen is displaying the first page you will be shown the last page

When you are finished looking at a given sector, you may instruct the program to show you another sector by pressing the f7 key. It will then ask you for the track and sector numbers and proceed as above. If you press the f4 key instead the program will dispense with asking you for a track and sector and will display the next block in the file.

If you wish to modify a sector, you may use the cursor control keys to move to the hex digit of the byte you wish to change, then press any valid hex digit key (0 to 9 or A to F). The ASCII on the right will also be changed to reflect the new value. You may change any number of bytes in a sector and freely flip pages with the f1 and f3 while doing so. When you have the sector looking the way you want it, press f8 to actually update the disk. Until you do this your changes are only kept in RAM, so if you make a mistake and want to get back to what was in the sector before, you need only press f7 and reread the same sector (or you could go on to some other sector and forget the whole thing).

When you are through monkeying around with your floppy, press f2 to exit the program. Note: Don't exit by pressing RUN/STOP as this will leave the disk channels open and terrible things may result.

If you press any key which the program cannot interpret, it will flash a help screen telling what the valid keys are and what they do. If you forget which function key does what, simply hit anything invalid, such as the return key or a question mark, and you will be shown this help screen.

PROGRAMMING NOTES

A few comments may be in order on some of the less comprehensible portions of this program.

The first requirement in writing it was to be able to get data off the disk by track and sector and then put it back after it was modified. This could have been difficult to do had not Commodore conveniently provided the U1 and U2 disk commands. The U1 command causes the drive to read a block specified by a track and sector number and load it into a buffer in the drive unit; U2 takes data from the buffer and writes it onto the disk. The only remaining problem was transferring data between the buffer and the computer itself. The normal way to read data is with INPUT# or GET#. INPUT# was ruled out because it makes too many assumptions about what the data will look like. It drops leading blanks, expects to see a carriage return to terminate the input stream, and assumes each variable will be terminated by a comma or colon. None of this could be expected to be true for an arbitrarily chosen block on the disk, especially for those containing directory information or programs. Thus GET# seemed the logical choice, and indeed that was what I used in my first version of the program. Unfortunately, this ran extremely slowly. Writing with PRINT#, while not as bad, was also unacceptably slow. Therefore, I decided to code the read and write routines in machine language. I have entered the POKE values in the listing with the corresponding assembler mnemonics as REMarks beside them as an aid to comprehension. Each routine transfers data from the buffer directly into or out of the integer array D%. Lines 1130 and 3130 are used to get the address of this array into locations 828-829 where it can be accessed by the ML routines.

Displaying the block is rather simple. Line 2150 prints the hex representations of bytes, four per line, while line 2160 prints the ASCII values. Some of these cannot be printed without messing up the display, most notably the color codes and cursor controls, and so are converted to periods.

Control of the cursor for modification is a bit unusual. Line 2220 computes which byte of the sector we're on from page, line, and column. It also computes an adjusted column number (C1) to allow for the space after every two hex digits on the display. Line 2225 sets up and calls the kernal plot routine. Line 2230 turns on the blinking cursor, and line 2250 shuts it off again after getting a character and then puts back whatever was in that position before (so we don't leave inverse video blocks scattered about the screen).

Whenever a byte is modified, the new value is computed in lines 2430-2450 by simply running the hex representation through the input routine. This is admittedly a bit clumsy, but it was short and easy to write and it seems to run sufficiently fast.

FINAL COMMENTS

Beginning users are cautioned against trying to modify the disk directly when they are not sure exactly what they are doing, especially with regard to the directory. A good idea might be to practice on a 'junk disk' with a few files tossed on it, to see exactly what happens.

Another caution to bear in mind is this: if you do anything to change which sectors are "in use," be sure to run the verify (V command) to update the Block Allocation Map. (This is a table which the system keeps on track 18, sector 0, in which it records which blocks are in use and which are available.) While it is possible to alter the BAM yourself, I definitely recommend against it. Its format is rather obtruse, making it easy to err, and any error can result in data being overwritten—Verify is easier and safer. ☐

SEE PROGRAM LISTING ON PAGE 105

SUPER DUPER

A Sequential, Program, and Machine Language File Copier for the 64

By Dave Barron

The *Super Duper* file copier utility will allow disk users to copy sequential or program files easily from one disk to another (or onto the same disk). It will also allow you to rename the file. One nice feature is that you can also copy machine language programs without having to know ending and starting addresses, thus making it easy to copy ML programs.

Super Duper demonstrates the use of what I like to describe as hybrid programs. These are programs I enjoy writing because I can take advantage of the easy to use BASIC language and combine it with the speed and/ or necessity of using machine depending upon the application. Further, by using kernal routines I can create short effective programs while letting the already built-in ML routines do much of the work.

A brief description of the BASIC program listing follows. Also included on this page is a disassembly of the actual ML code that is POKEd in by the BASIC program, for those that wish to follow the ML logic, with brief comments. □

Line #s	Description
10- 30	Clear the screen and POKE in the ML code from the data contained in lines 240 through 280. The numbers POKEd in by line 30 set memory; pointers used for keeping track of where we are in memory later on and initializing a counter location.
40- 80	Print onscreen menu.
90-110	Get user's choice, convert his string response to a numeric value, and jump to appropriate line to handle choice.
120-150	Allow user to input file name.
160-180	Open disk file and jump to ML routine beginning at 3584, which reads file into computer's memory.
190	Sets where end of file just read in exists, reset memory allocation, and reinitialize counter.
200-210	Get user's new filename and allow user to swap disks.
220-230	Open disk file to write file to and jump to ML routine at 3621 to write file to disk from computer's memory.
230	Tells user copy is complete and goes back to see if he wants to copy any more files.
240-280	The data statements with the actual ML code.
290	End program.

SEE PROGRAM LISTING ON PAGE 107

ML Code Disassembly

```
3584 A2 03    LDX #$03     ;load X reg with file #3
3586 20 C6 FF JSR $FFC6    ;JMP to CHKIN kernal routine
3589 20 E4 FF JSR $FFE4    ;JMP to GETIN kernal routine and put
                            character in A reg
3592 AC 3C 03 LDY $033C    ;load Y reg with counter
3595 91 FB    STA ($FB),Y  ;put character gotten into memory
3597 C8       INY          ;increment Y reg (the counter)
3598 8C 3C 03 STY $033C    ;store the counter into counter memory
                           location
3601 C0 00    CPY #$00     ;compare counter to zero
3603 D0 08    BNE $0E1D    ;if not 0 branch to 0E1D
3605 E6 FC    INC $FC      ;if counter was 0 increment memory
                            pointer
3607 A6 FC    LDX $FC      ;load X with pointer value
3609 E0 9F    CPX #$9F     ;is file too large to fit into memory
3611 F0 04    BEQ $0E21    ;if file too large then branch to 0E21
3613 A5 90    LDA $90      ;load A reg with status word
3615 F0 E4    BEQ $0E05    ;if it's 0 branch to 0E05, get another
                            character
3617 20 CC FF JSR $FFCC    ;JMP to CLRCHN kernal routine and clear I/O
                            channels
3620 60       RTS          ;return to BASIC
3621 A2 03    LDX #$03     ;load X reg with logical file #3
3623 20 C9 FF JSR $FFC9    ;JMP to CHKOUT kernal routine and open channel
                            for output
3626 AC 3C 03 LDY $033C    ;load Y reg with counter
3629 B1 FB    LDA ($FB),Y  ;load A reg with first character from
                            memory
3631 C4 FD    CPY $FD      ;compare Y reg to what's in pointer
                            location
3633 D0 06    BNE $0E39    ;if not equal branch to 0E39
3635 A6 FC    LDX $FC      ;load X reg with what's in pointer
3637 E4 FE    CPX $FE      ;compare what's in X to what's in FE
3639 F0 10    BEQ $0E49    ;if it's equal branch to 0E49
3641 C8       INY          ;increment Y reg
3642 8C 3C 03 STY $033C    ;store Y reg into counter location
3645 20 D2 FF JSR $FFD2    ;JMP to CHROUT kernal routine and output
                            character to disk
3648 C0 00    CPY #$00     ;compare Y reg to 0
3650 D0 E6    BNE $0E2A    ;if not 0 branch to 0E2A & get another
                           character
3652 E6 FC    INC $FC      ;increment pointer
3654 4C 2A 0E JMP $0E2A    ;go back and get another character
3657 20 CC FF JSR $FFCC    ;JMP to CLRCHN kernal routine and clear I/O
                            channels
3660 60       RTS          ;return to BASIC
```

TWO-COLUMN DIRECTORY FOR THE C-64

BY RICHARD LOVETT AND BARRY ARCHER

Have you ever listed a disk directory, only to have half of it scroll out of sight before you could read it? Have you ever wanted to load or erase files right from the directory without typing all those lengthy commands? Then *Two-Column Directory* is for you.

As its name implies, *Two-Column Directory* displays the directory in two rows, making it possible to view 40 file names at once instead of the usual 25 or fewer. (Many of your disks probably contain fewer than 40 files, so much of the time you'll be able to see the full directory in one shot.)

The program also allows you to load or erase files by putting the cursor on the file name of your choice and pressing just one or two keys.

In addition to working as a stand-alone program, *Two-Column Directory* can be pared down to a subroutine for use within your own programs. The subroutine that we will describe omits the load and erase features in order to make the length of the code manageable. The resulting routine will provide a two-column directory from within your program without erasing your program in the process.

In order to display dual rows of directory information, *Two-Column Directory* had to compromise somewhere. The Commodore 64's 40-column screen can't show two file names side by side and still include the program types and number of blocks occupied by each file. So, *Two-Column Directory* displays the program type as one letter (P for Program, S for Sequential, and R for Relative) and omits the file size. Most of the time, the size is not crucial anyway.

After listing the first 40 file names (or fewer, if that's all the disk contains), the program beeps, displays a rectangular yellow cursor, and shows a command line at the bottom of the screen. This line reminds you that f1 will continue the listing, f3 will start the listing over again, f5 loads a program, and f7 erases one.

To load or erase, use the cursor control keys to put the cursor on top of the file name of your choice. *Two-Column Directory* is set up to load any program file that uses the LOAD "filename",8 syntax. If you press f7 to erase, the command line asks you to confirm your intent before it scratches the file. And, if you accidentally put the cursor on a blank line before pressing f5 or f7, the program ends.

Assuming there are more than 40 files on the disk, a press of f1 causes *Two-Column Directory* to erase the lefthand column and show the next 20 file names; if still more remain, another press of f1 erases the righthand column and shows an additional 20. This procedure continues until the full directory has been displayed.

To trim things down to a manageable subroutine, omit lines 70, 140-160, 200, 210, 250, and 260-580. Then make these additions or substitutions:

```
•25 COL=1
•140 IF BY$="" OR BY$=CHR$(34) THEN BY$=C
 HR$(32)
•165 PRINT BY$;: NEXT
•205 IF COL=2 THEN PRINT HO$+"[DOWN]":GOT
 O 110
•210 PRINT "[DOWN][12" "][RVSON] PRESS A
 KEY"
•211 GET J$: IF J$="" THEN 211
•212 PRINT HO$+"[DOWN][DOWN]";
•225 PRINT SP$;SP$;HO$;"[DOWN]": GOTO 110
```

You will also need to add either a RETURN or GOTO line following line 255 in order to send control back to your main program. ☐

SEE PROGRAM LISTING ON PAGE 103

at TUSSEY COMPUTER PRODUCTS*. . .

WE KNOW WHAT YOU WANT!

We take a bite out of prices!

WE VALUE YOUR BUSINESS. IN ORDER TO EARN IT, WE ARE GOING TO GIVE YOU THE THINGS YOU WANT MOST FROM A MAIL ORDER COMPUTER STORE.

1) **Low Prices!** We try to have the lowest prices around. If you do find a lower price, please give us a chance to beat it.
2) **Fast, dependable service** - Our shipping department works hard to get your orders out quickly, usually within one business day.
3) **Straight talk** - We will give you accurate information as to our stocking condition.
4) **Personnel who care** - The people at TCP sincerely want you to be happy with the way your order is handled. We all know our jobs depend on having a happy customer.

ORDERS ONLY PLEASE: 800-468-9044

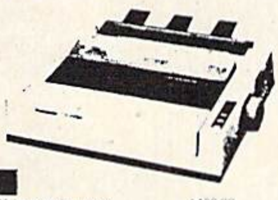
INFORMATION & PA ORDERS: 814-234-2236

PHONE LINES OPEN 10-8 MON-FRI, 10-5 SAT

THE PANASONIC KXP 1091 PRINTER

Near letter quality at the flick of a switch

The 1091 zips along at 120 cps and has a 1k buffer. It also has all the print modes of the Gemini 10X and Epson RX-80, plus a BIG difference. The 1091 has a near letter quality switch. Flip it and you get a super tight 12 by 18 matrix with characters that rival a daisy wheel! The 1091 has both friction and tractor feed.



Our price $279.00

Panasonic KXP 1090 .... $199.00

(does not have near letter quality)

SG 10, NLQ Dot Matrix from Star ...... Call

PRINTERS

Gemini 10X	$234.00*
SG 15	Call
Delta 10, 160 cps, 8k buffer	$389.00*
Delta 15	$580.00*
Powertype, 18 cps, letter quality	$329.00*
Cardco LQ1 letter quality printer	$429.00
Cardco LQ3	$319.00
Epson RX-80	$244.00
Epson Printers	Call
Okidata Printers	Call
Hush 80	$109.00

*Star Micronics printers come with a 1 year warranty!

PRINTER INTERFACES

Xetec GPI	$ 76.99
Xetec SPI/B	$ 59.99
Xetec SPI	$ 43.99
Cardco ?/+G	$ 67.00
Cardco ?/B	$ 44.00
Cardco PS Interface	Call

DATABASES

The Consultant	$ 63.99
Mirage Database Manager/ Report Generator	$ 69.99
Superbase 64(d)	$ 56.99
Data Manager II by Timeworks	$ 32.99
Practifile by MS(d)	$ 37.00

MODEMS

Total Telecom (1650 compatible)	$ 76.95
Commodore 1660 New!	Call
Compuserve starter kit	$ 26.95
Westridge modem	$ 76.95
Mighty Mo, new modem from USI	$ 78.99
VIP terminal package by Softlaw	$ 39.95
Vidtex Terminal	$ 30.00
Vidtex plus Compuserve Starter	$ 49.99

WORD PROCESSORS

Word Processor by Proline(d)	$ 39.99
Wordpro 3+/64 w/spellright(d)	$ 59.00
Paperclip(d)	$ 59.00
Paperclip w/spellpack(d)	$ 76.99
Cardco Write Now/64 (cart)	$ 37.00
Mirage Professional W.P. (d)	$ 59.00
Omni Writer/Speller(d)	$ 45.00
Word Writer(d) by Timeworks	$ 32.99
Script 64	$ 39.99

CARDCO

Light pen	$ 29.00
Numeric keypad	$ 35.00
5 slot exp. interface CB/5	$ 58.00
Mail Now/64	$ 32.00
Spell Now	$ 31.99
File Now	Call
Graph Now	Call

HARDWARE

1541 Express by RTC	$ 73.99
Smart Cable	$ 49.95
BI-80 Batteries Including 80 column card with Basic 4.0 built in	$137.99
BusCard II by Batteries Included	Call

SPREADSHEETS

Calc Result Adv. (d, cart)	$ 67.00
Calc Result Easy (cart)	$ 33.99
P.S. (Prog. Spreadsheet)(d)	$ 49.99
Practicalc 64(d)	$ 36.00
Practicalc 64(t)	$ 34.00

MONITORS

Zenith 12" Amber	$ 97.00
Zenith 12" Green	$ 92.00
Sakata SC 100 Color Monitor	$239.00
Amdek Color 1 Plus	$259.00
Cable for monitors	$ 9.95
1702	Call

UTILITIES

Pal 64 assembler by Proline(d)	$ 39.99
Power 64 Basic by Proline(d)	$ 39.99
Toolbox 64, both Pal 64 and Power 64(d)	$ 69.99
Canada A/M backup program	$ 39.95
Simon's Basic	$ 39.95
CSM 1541 Alignment	$ 32.99

MISCELLANEOUS

Koalpad w/painter(d)	$ 65.00
Spellpro 64(d) by Proline	$ 39.99
Mailpro 64	$ 39.99
Complete Personal Accountant(d)	$ 54.00
The Home Accountant	$ 36.95
Timeworks Inventory, A/P, A/R, Cash Flow Management, General Ledger, Payroll Management	$40.95 each
64 Docter	$ 24.95
FCM, First Class Mail	$ 36.95
Super Sketch	$ 39.99

DISK DRIVES

MSD SDI	$249.00
MSD SDII	Call
Indus GT Disk Drive	Call
1541	Call

★ NEW PRODUCTS ★

Trio from Softsync, Integrated W.P., Database, Spreadsheet	Call
Homepak from Bat. Incl., Integrated W.P., Database, Telecommunications	Call
Vizastar Int. Spreadsheet, Database, Business Graphics	Call
Epyx Fast Load	Call
Broderbund Print Shop	$ 31.99

Please Note: When we ask you to call for a price it's because prices on those items are volatile, and we want to be sure we can give you the lowest prices we can.

To Order By Mail: Send personal check/money order or certified check. Allow 2 weeks for personal checks to clear.

Shipping: Software and accessories add $2.50 per order. COD order add $6.00 (cod's not accepted on printers, monitors, and disk drives.) -Printers add $10.00. Disk Drives and Monitors add $8.00. Call for other shipping charges. Additional shipping required on APO, FPO, AK, HI, and foreign orders.

Terms: Orders shipped UPS unless noted otherwise. All prices reflect cash discount. Add 3% for Visa and Mastercard. Manufacturer's warranty honored with our invoice and original packaging. PA residents add 6% sales tax. Prices subject to change. All sales final. Defective products replaced or repaired at our descretion.

"DEALER INQUIRIES INVITED"

Other product lines available! Call for prices.

commodore 64

NEW PRICES!

Commodore 64 is $147.00*

*With the purchase of any printer, disk drive, monitor, or modem.

1541 Disk Drive	$195.00
1702 Color Monitor	$195.00
1660 Modem	Call
Plus 4, C-16, others	Call
SX 64	$495.00

NEW PRINTER FROM STAR!

The SG-10 is an upgrade of the popular Gemini 10X. It runs at 120 CPS, and has all the print modes of the 10X. What makes this printer different is a near letter quality mode at 60 CPS, with a 17 by 11 matrix! With this mode the printer does double duty, eliminating a need for a letter quality printer. It also includes a built in 2k buffer, and a 1 year warranty.

PLEASE CALL FOR OUR LOW PRICE!

BUY A BUNDLE— $AVE A BUNDLE

Order one of these popular packages of products by number for super savings!

Package #5
WORD PROCESSOR 64 and SPELLPRO 64
The word processor and spelling checker from Proline Software
$73.99

Package #6
1541 EXPRESS and the FAST CABLE
Everything you need to speed up your disk drive
$119.99

Package #7
CONSULTANT and PAPERCLIP W/SPELLPACK
These are the popular Database and Word Processor from Batteries Included
$129.99

Package #9
MIRAGE DATABASE/ADVANCED REPORT GENERATOR and MIRAGE PROFESSIONAL WORD PROCESSOR
A powerful package that integrates together
$124.95

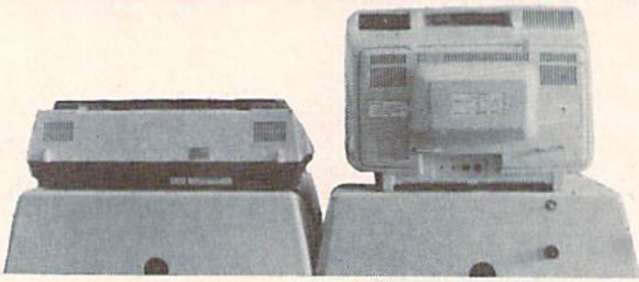
Package #10
BI-80 and PAPERCLIP W/SPELLPACK
The 80 column card with the Basic 4.0 built in, and the Paperclip w/spellpack both from Batteries Included. Paperclip works with the 80 column card to give you a powerful word processor
$209.95

TUSSEY COMPUTER PRODUCTS
* (formerly Tussey Mountain Software)
BOX 1006
STATE COLLEGE, PA 16804

INNOVATIVE ORGANIZERS, INC.

For Commodore 64™ & VIC-20™





WELCOME, to the family of **INNOVATIVE ORGANIZERS.** These **ATTRACTIVE** power organizer stands, made of **ABS PLASTICS, U.L. RATED,** provide an opportunity for a more **PROFESSIONAL** looking computer installation, both at **HOME** and in the **WORK PLACE.**

There is **AMPLE** room under the organizer to place all the computer accessories: **POWER SUPPLY, MODEM, GAME CARTRIDGES, MEMORY EXPANSION,** and **MUCH MORE.**

Now, let's examine how these organizers work. There are 3 CRT-Models.

CRT-1 holds a Monitor or T.V. Set and a VIC 20 or 64. All wires & cartridges fit neatly under the organizer.

CRT-2 same as CRT-1, but also has: **ONE-STEP MASTER SWITCH** (permits 4 pieces of hardware to be turned on and off from one switch, plus circuit protection, incoming power fused with 6 AMP Fuse).

CRT-3 same as CRT-2, but also has: **SURGE & SPIKE PROTECTION** built in.

P-D-1: **HOLDS 1525 PRINTER & 1541 DISK DRIVE.** All wires & cartridges fit neatly under the organizer.

Ask about our Power Lock Switch & Power Portable Computer Fan & 2 Outlet Surge & Spike Protection.

I.O.

YOUR ALTERNATIVE

INNOVATIVE ORGANIZERS

P.O. Box 108, Markleville, IN 46056

Reader Service No. 208

CRT-1	$17.98
CRT-2	$29.97
CRT-3	$37.97
P-D-1	$19.48

Optional Accessories

SS-1 Surge & Spike Suppressor $29.95

CF-1 Computer Fan $29.95

Plus Shipping

CALL TOLL FREE
1-800-233-3645 Ext. 15

Disk Spinners

Replacements and Enhancements for the 1541 Disk Drive

By Morton Kevelson

Last month we examined the Indus GT and Commander II drives, 1541 Physical Exam, and On Track Indicator. This month we conclude with Kwik-Load!, 1541 Flash, and Fast Load.

PART II



READER SERVICE NO. 175



READER SERVICE NO. 176



READER SERVICE NO. 177

KWIK-LOAD!

Datamost Inc.
20660 Nordhoff Street
Chatsworth, CA 91311
Phone: 818-709-1202
Price: $19.95

While placing the emphasis on fast disk operation, the *Kwik-Load!* name neglects to mention the additional features of this low-cost yet powerful disk utility. Written by Mark Riley, author of *Paint Magic*, the *Kwik-Load!* utility disk contains two programs. The first speeds up the LOAD time of the 1541 with the Commodore 64 by a factor of five. The second is a collection of disk utilities which simplifies and speeds up the most common disk file maintenance operations.

The *Kwik-Load!* utility is a short machine language routine which loads into the $C000 block in the Commodore 64. It is linked into the C-64 operating system by changing the LOAD vector at $330-$331 to point at itself, thus diverting all subsequent LOADs away from the usual Kernal routines. Although a three to one speed benefit is touted by the program's purveyors, the actual benefit is a combined function of the type of disk activity, program length, and number of files being accessed. Typical speed improvements hovered around the five to one mark. For example, *Paper Clip*, our perennial favorite for long LOADs, clocked in at less than 15 seconds, down from over a minute and a quarter on the first try. Note that *Kwik-Load!* only works when loading programs. It has no effect on SAVE times.

As with all RAM based speedup utilities, *Kwik-Load!* will perform best with BASIC programs. Most autoboot commercial software will disconnct the program during the boot process. A second version of *Kwik-Load!*, supplied on the disk, LOADs into the top of free BASIC RAM. In the process, it reduces the amount of available memory for BASIC programs by one kilobyte. In exchange, Commodore's DOS wedge can be LOADed in the usual fashion.

KWIK-COPY

Kwik-Copy, the second part of this package, is a comprehensive disk utility program. The highlight of the package is a high speed disk copy routine which will copy an entire disk in 4.5 minutes and three passes with just a single disk drive. If two drives are used, the process takes a mere 3 minutes 40 seconds (of course the need to swap disks back and forth is also eliminated). These times were obtained with the verify option turned off. Turning verify on will slow things down a bit, but not very much. The routine aborts on encountering a bad sector, so do not wait for those disks to start going bad before backing up.

Kwik-Copy automatically accounts for a second drive. If you have a hardware device nine on one drive, turn them both on and boot the program. If both drives are a device eight, turn one on and boot up. When done, turn on the second drive.

Kwik-Load! will automatically sense the presence of the additional drive and perform a soft change of the device number.

Other utilities included are an easy to use sector editor with a rather nice two part hex and ASCII display, a high speed file copy utility, and a BAM directed copy routine. A drive speed checking routine is included as well. The speed test is completely passive as it does not write to the disk surface when in operation.

The file copy routine has a very handy feature which the manual fails to document properly. When the routine prompts for a file name, enter the asterisk (*) character as a default. The program will automatically display each file name and ask if you wish to copy it.

CONCLUSION

Kwik-Load! is the first of a series of competitively priced ($19.95) *Kwik-Ware!* software to be released by Datamost. All of these programs will incorporate the same high speed disk routines. It is a package well worth consideration by the budget conscious. □

1541 FLASH

Skyles Electric Works
231 E. South Whisman Road
Mountain View, CA 94041
Phone: 415-965-1735
Price: $89.95

The Skyles Electric Works 1541 Flash consists of two eight kilobyte ROMs with associated hardware and a disk drive to computer connecting cable. The Flash is actually a replacement for the C-64 Kernel and half of the 1541 DOS. In operation, the Flash provides approximately a three to one speed improvement for all disk operations. A mini DOS wedge and some C-64 program editing support are included as well.

HARDWARE AND INSTALLATION

The 1541 Flash consists of several components. In addition to the two ROMs mentioned above, there is a pair of printed circuit adapter boards, a cable assembly, and a combination connector/switchboard/user port extension assembly.

Installation of the Flash is not for everyone. At best it will require disassembly of both the Commodore 64 and the 1541 disk drive. At worst, considerable adaptive surgery of the C-64 circuit board will have to be performed.

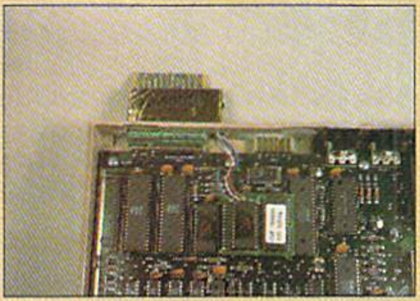
On the C-64 side you will have to remove the Kernal ROM and replace it with an adapter board which allows for the simultaneous installation of two ROM chips. This will present little difficulty if your C-64 has its Kernal ROM in a socket. Many of the more recent editions of the C-64 have the chips permanently soldered into place. Extraction, in this case, is a tricky procedure involving the simultaneous unsoldering of a chip with 24 leads. This has to be done without damaging either the circuit board or the original Kernal ROM. If your C-64 has a soldered-in Kernal and your dealer is not able to perform the necessary microsurgery (a likely occurrence), Skyles will do the service for $28.50. Given round trip transportation for the computer, this would probably be a three week minimum proposition. If you happen to be in northern California with your C-64, you may call for an appointment for an on the spot replacement.



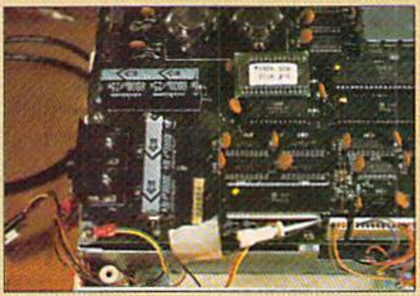
Startup screen with Flash installed.



Flash hardware components (see text).



Flash installed in C-64's user port.



Flash installed in 1541 disk drive.

A special adapter board replaces the original ROM chip. This consists of the replacement ROM, a second socket to allow reinstallation of the original ROM, and a pair of wires which go to a second adapter board which plugs into the user port.

The second board serves several purposes. It includes a small switch which chooses between the two ROMs now installed in the C-64. It provides a convenient termination point for the extra cable which goes to the 1541 disk drive. It connects up to two of the lines on the CIA port and it extends the user port itself for use by other accessories.

As a result, the user port contacts are fully exposed to possible electrical contact with other objects. An insulated protective cover should have been provided with the kit. Also, unlike the original C-64 contacts, the extension board is not gold plated.

In the 1541, one of the operating system ROMs has to be pried from its socket and replaced. Also, one of the 40 pin VIA chips is yanked, a pin is bent so it points away from the body of the chip, and the chip is reinstalled. A pair of micro chips on the supplied cable assembly are connected to the VIA chip. Routing of the cable completes the installation.

We felt that an adapter board with a permanent connection would have done away with the need for the microchips or any bending of chip leads. For our test installation, we rigged a 40 pin socket to eliminate the lead bending step.

OPERATION

Bryce Nesbitt is a name with which owners of the Flash will become intimately familiar. The C-64's turn on greeting message is augmented to remind you of Bryce's handiwork whenever you power up. User RAM is unaffected by the Flash. The usual 38911 bytes free appears. The switch on the user port adapter allows you to completely restore the original configuration of the C-64 by electrically switching between the original ROM and the Flash ROM. This will allow use of the computer with an unmodified disk drive or with any user port accessories which may require all of the CIA lines. The hardware change in the 1541 is irreversible. A DOS command has been added to provide a soft reset of the disk drive to permit operation with the unmodified C-64.

In operation, we found the Flash to give an approximately three to one speed improvement for all disk drive operations. This included file operations as well as program loads. The actual speed benefit varied with the particular sequence of operations. For example, *Easy Script* went from 62 to 30 seconds, *Paper Clip* from 78 to 23 seconds, and *Standing Stones* by Electronic Arts from 139 to 80 seconds. Programs which perform complex operations on multiple levels, typically databases, will also benefit from the much faster head movement which accompanies the Flash in the 1541. This high speed head movement remains in effect even when the Flash is deactivated.

COMPATIBILITY

Compatibility with commercial software is quite good. All the standard Commodore DOS commands are recognized. Most copy protected software worked without any problems, although some programs would load only if the Flash was turned off. We did come across some samples which would not work at all with the Flash-equipped disk drive. An arrangement for the disk drive similar to the C-64, where the original ROM can be switched back in, would have been beneficial.

SYSTEM ENHANCEMENTS

The Flash includes a tiny DOS wedge which simplifies issuing disk commands and reading the error channel. The commercial at (@) symbol replaces the OPEN 15,8,15: PRINT#15, sequence when followed by any standard DOS command. The character alone reads the error channel. The disk directory can be displayed to the screen without affecting the program in memory.

The Commodore full screen editor is enhanced with several new features. These include rapid cursor movement to the bottom of the screen, a sixteen character tab, line delete, partial screen clear, and a program LISTing pause.

Several DOS commands have been added as well. These deal with the block read and write operations and are intended for programmers who wish to directly utilize the high speed Flash routines.

CONCLUSIONS

The 1541 Flash does make working with the 1541 disk drive far more comfortable. The waiting time for long files to LOAD, which used to be an everlasting tedium, has been reduced to a tolerable bore. Shorter LOADs which merely involved an annoying delay respond with a welcome snappiness. The fairly complex installation will deter many users from the purchase of a Flash. We do suggest you check your C-64 for a socketed Kernal ROM before making a final decision. The manual does include a detailed installation section, with reasonably clear photographs, which should prove adequate for anyone with some familiarity with the hardware.

Overall, the Flash was a pleasure to work with. It was not without some reluctance that we removed it from our system after the thirty day evaluation period. ☐

FAST LOAD

Epyx
1043 Kiel Court
Sunnyvale, CA 94089
Phone: 418-745-0700
Price: $40.00

The Epyx *Fast Load* utility is packaged as an eight kilobyte ROM in a plug-in cartridge. You just plug it in to the cartridge port of the C-64 and forget about it. The most dramatic improvements in disk drive speed are associated with the LOADing of long program files, the longer the better. For example, *Paper Clip* normally LOADs in one minute and eighteen seconds; with *Fast Load* it only takes fifteen seconds—a more than five to one speed up.

The performance is not nearly as impressive when you get away from straight LOAD operations. For example, *Standing Stones* from Electronic Arts normally boots in two minutes and twenty seconds. With *Fast Load* the procedure is reduced to one minute and forty seconds, a forty percent increase in speed. Another example is Commodore's *Easy Script*, where the original LOAD time of one minute and two seconds showed a mere five percent improvement at fifty nine seconds. Of course, other cartridge-based software was actually slowed up by the amount of time it takes to yank *Fast Load* from the expansion port. Finally, working from within *Easy Script*, we found no discernible difference in the time which it took to read and write text files.

A DOS WEDGE—BUILT IN

Fast Load offers far more than a five to one speed increase in program LOADs. It is actually several utilities in one package plus a number of DOS convenience features thrown in for good measure. When your C-64 is powered up with the *Fast Load* cartridge in place, the only discern-

ible difference is the product name along with the usual sign-on message. Do not let this lull you into a false sense of smug somnambulance. Lurking at your fingertips are a myriad of hitherto unavailable features just awaiting your beck and call.

The first and foremost is an instant DOS wedge. To get a directory listing just type "$" (that's SHIFT 4 for all you neophytes) and hit RETURN. *Voila*, the directory scrolls by right before your eyes. Furthermore, any program residing in the computer's memory remains totally oblivious to the turmoil taking place around it. The only drawback was that there seemed to be no way to slow down or put a halt to the process. Once started, the directory seemed most determined to scroll by to its final conclusion where it profoundly proclaims the disk BLOCKS FREE message.

Hitting RUN/STOP and RESTORE just made matters worse. The screen cleared and the drive went off to sulk, apparently on an endless binge of disk spinning. Even depriving it of its orbiting celluloid had no noticeable effect. Despair not; the commercial at (@) is available for inquiring into the health and well-being of the disk drive as it reads the DOS error channel. Exercising its function brings a response of blissful contentment (a display of 00 OK 00 00) shortly followed by a cessation of all rotary motion.

One other peculiarity commanded our attention. The traditional means of querying the disk contents, namely LOAD "$",8, returned a gaggle of worthless gibberish. However, the slash (/) character, which is now available in addition to the traditional LOAD command, served well in the classical sense. The directory thus LOADed retains its useful meaning. Other handy characters are the percent (%) symbol which replaces the binary LOAD "FILE",8,1 sequence and the left arrow (←) which replaces the BASIC SAVE.

Last but not least, the well-known LOAD "*",8,1 sequence, used by many purveyors of commercial software, is now available by the simultaneous depression of the Commodore logo and RUN/STOP keys.

THE DISK TOOL

In addition to the built-in Wedge, *Fast Load* has a series of utilities referred to as the *Disk Tool*. Upon critical analysis, we would consider these a full-fledged crowbar. The British Pound key is used to access the *Tool* utilities. These are most comprehensive. Included are:

- A single file copy utility which is limited to program files.
- A full disk copy utility which allows for preformatted as well as post-formatted disks. This routine takes about ten minutes.
- A BAM-directed disk copy utility. This will only copy sectors which are marked as allocated on the block availability map.
- A disk sector editor utility. This allows you to read any disk sector,

A Printer For All Reasons

Search For The Best High Quality Graphic Printer

If you have been looking very long, you have probably discovered that there are just too many claims and counterclaims in the printer market today. There are printers that have some of the features you want, but do not have others. Some features you probably don't care about; others are vitally important to you. We understand. In fact, not long ago, we were in the same position. Deluged by claims and counterclaims. Overburdened by rows and rows of specifications, we decided to separate all the facts — prove or disprove all the claims to our own satisfaction. So we bought printers. We bought samples of all major brands and tested them.

Our Objective Was Simple

We wanted to find that printer which had all the features you could want and yet be sold directly to you at the lowest price. We wanted to give our customers the best printer on the market today at a bargain price.

The Results Are In

The search is over. We hae reduced the field to a single printer that meets all our goals (and more). The printer is the GP-550 from Seikosha, a division of Seiko. We ran this printer through our battery of tests and it came out shining. This printer *can* do it all. Standard draft printing up to a respectable (and honest) 86 characters per second, and with a very readable 9 (horizontal) by 8 (vertical) character matrix. At this rate, you will get an average 30 line letter printed in only 28 seconds.

"NLQ" Mode

One of our highest concerns was about print quality and readability. The GP-550 has a print mode termed Near Letter Quality printing (NLQ mode). This is where the GP-550 outshines all the competition. Hands down! The character matrix in NLQ mode is a very dense 9 (horizontal) by 16 (vertical). This equates to 14,400 addressable dots per square inch. Now we're talking *quality* printing. You can even do graphics in the high resolution mode. The results are the best we've ever seen. The only other printers currently available having resolution this high go for $500 and more *without* the interface or cable needed to hook up to your computer.

Features That Won't Quit

With the GP-550 your computer can now print 40, 48, 68, 80, 96, or 136 characters per line. You can print in ANY of 18 font styles. You not only have the standard Pica, Elite, Condensed and Italics, but also true Superscripts and Subscripts. Never again will you have to worry about how to print H_2O or X^2. This fantastic machine will do it automatically, through easy software commands right from your keyboard. All fonts have true descenders.

One of the fonts we like best is "Proportional" because it looks most like typesetting. The spacing for thin characters like "i" and "l" are given less space which "tightens" the word making reading easier and faster. This is only one example of the careful planning put into the GP-550.



Do you sometimes want to emphasize a word? It's easy, just use **bold** (double strike) to make the words stand out. Or, if you wish to be even more emphatic, underline the words. Or do **both.** You may also wish to "headline" a title. Each basic font has a corresponding elongated (double-wide) version. You can combine any of these modes to make the variation almost endless. Do you wnat to express something that you can't do with words? Use graphics with your text — even on the same line.

You can now do virtually any line spacing you want. You may select 6, 8, 7½ or 12 lines per inch. PLUS you have variable line spacing of 1.2 lines per inch to infinity (no space at all) and 97 other software selectable settings in between. You control line spacing on a dot-by-dot basis. If you've ever had a letter or other document that was just a few lines too long to fit a page, you can see how handy this feature is. Simply reduce the line spacing slightly and ... VOILA! The letter now fits on one page.

Forms? Yes! Your Letterhead? Of Course!

Do you print forms? No problem. This unit will do them all. Any form up to 10 inches wide. The tractors are adjustable from 4½ to 10 inches. Yes, you can also use single sheets. Plain typing paper, your letterhead, short memo forms, anything you choose. Any size under 10" in width. Multiple copies? Absolutely! Put forms or individual sheets with carbons (up to 3 deep), and the last copy will be as readable as the first. Spread sheets with many columns? Of course! Just go to condensed mode printing and print a full 136 columns wide. Forget expensive wide-carriage printers and changing to wide carriage paper. You can no do it all on a standard 8½" page.

Consistent Print Quality

Most printers have a continuous loop ribbon cartridge or a single spool ribbon which gives nice dark printing when new, but quickly starts to fade after a while. To keep the printers' output looking consistently dark, the ribbons must be changed more often than is healthy for the pocketbok. The GP-550 solves this problem completely by using a replaceable, inexpensive ink cassette which is separately replaceable from the actual ribbon. It keeps the ribbon loaded with ink at all times. You only replace the ribbon when it truly wears out, not when it starts to run low on ink. Just another example of the superb engineering applied to the GP-550. (When you finally do wear out your ribbon, replacement cost is only **$10.95**. Ink cassette replacement cost is only **$5.95**, both postpaid.)

Reader Service No. 190

The Best Part

When shopping for a quality printer with all these features, you could expect to pay around $500 or more. *Not any more!* We have done our homework. You don't have to worry about interfaces or cables. Everything is included. You need absolutely nothing else to start printing — just add paper.

No Risk Offer

We give you a 15-day satisfaction guarantee. If you are not completely satisfied for any reason we will refund the full purchase price. A 1-year warranty is included with your printer.

The Bottom Dollar

GP-550A	Standard Parallel (No Cable)	$249.95
GP-550CD	Commodore (Direct Connect)	$259.95
GP-550AT	Atari (Direct Connect)	$259.95
GP-550AP	Apple II or IIe (Direct Connect)	$299.95
GP-550PC	IBM PC & Compatables (No Cable)	$259.95
GP-550TI	TI 99/4A (Direct Connect)	$299.95

"Prices & Availability Subject to Change. CALL!"

Shipping is $8.00 — UPS within the continental USA. If you are in a hurry, UPS Blue (second day air) is $18.00. Canada, Alaska, Mexico are $25.00 (air). Other foreign is $60.00 (air). California residents add 6% tax. These are cash prices — VISA and MC add 3% to total. We ship the next business day on money orders, cashiers' checks, and charge cards. A 14-day clearing period is required for checks.

TO ORDER CALL TOLL FREE
1-(800) 962-5800 USA
OR 1-(800) 962-3800 CALIF.

or send payment to:

APROPOS TECHNOLOGY
1071-A Avenida Acaso
Camarillo, CA 93010

Technical Info: 1-(805) 482-3604

modify it, and write it back to any sector on the disk.

• A file utility which allows for locking and unlocking files (prevents inadvertant scratching), and renaming and copying of program files.

Overall, a rather nice collection of utilities to have in one place.

ML MONITOR

The *Fast Load* cartridge includes a machine language monitor. This program provides a full range of machine code disassembly and memory display modes, access to the disk directory and command channel, and both relocating and non-relocating binary LOADs. A full array of logical and arithmetic operators are available for easy calculation of pertinent parameters. Hexadecimal, decimal, binary, and ASCII conversion are built in. About the only thing lacking is the customary single-line mini assembler.

A COUPLE OF QUIRKS

We noticed some additional oddities worth passing on to the potential user. Executing a call to the C-64 reset vector (SYS 64738) restores the C-64 to the power-up state as if the *Fast Load* cartridge were not there. Resetting via a hardware reset button brings the *Fast Load* cartridge back online. The *Fast Load* cartridge may also be disabled via a command from the built-in *Disk Tool* utility.

A GOOD RETURN

The *Fast Load* cartridge offers a wide array of features in a compact package. The operation is surprisingly transparent to all the forms of commercial copy protection we have looked at. For the few exceptions, *Fast Load* may be totally disabled via the built-in soft switch without physically removing the cartridge. Overall, it may not be the fastest disk speedup utility, but it certainly offers a good return for the dollar. ☐

CONCLUSIONS

Of the three products we looked at, the Skyles Flash should provide the greatest overall improvement in disk handling with the 1541 disk drive. This is a direct result of applying the speedup routines to all disk operations under all operating conditions. If you can handle the relatively complex installation, then the Flash will significantly cut down on time spent waiting for the drive.

In terms of greatest convenience and utility, we place our bets on the Epyx Fast Load. The cartridge format makes for fast and foolproof installation as well as portability. As with the Flash, it is instantly available on power up. It also has the widest selection of built-in support utilities. Advanced users will be pleased by the instant availability of the powerful ML monitor.

The *Kwik-Load!* package offers a good value. The disk copy and backup utilities win the prize for speed. Given the low cost, you may just not want to pass this one up. ☐

EPILOGUE

Not being content to leave well enough alone, we present some last-minute updates on the Skyles Flash and the Epyx Fast Load cartridge. This information came to us just before we went to press on this issue.

THE 1541 FLASH

Skyles has indicated that the latest version of the Flash will include a defeat switch for the disk drive as well as the computer. This allows the Flash to be completely disconnected from the system. The price of the new unit will be $99. Present users may upgrade for $15 plus $3.50 shipping and handling. Work is nearly complete on a two disk version of the Flash which will sell for $139.95 ($49.95 for the upgrade). We hope to report on this when it comes available.

FAST LOAD

A second sample of the Epyx Fast Load cartridge appeared unannounced upon our doorstep. Although all external indications appeared identical to our first sample, we gave it a try. Much to our surprise, we found some subtle improvements. This latest revision properly handles the LOAD"$",8 command we discussed earlier. Epyx is apparently busy swatting bugs as fast as they find out about them. This latest version of Fast Load brings the count up to three revisions which we know about. At this rate, the product should be well in hand by the time you read this report.

DOS PLUS

A DISK OPERATING SYSTEM ENHANCEMENT UTILITY

The 1541 (or compatible) disk drive offers significant advantages over cassette as a storage medium, random access and speed of access being but two such. When you first purchased your drive unit you most likely realized a leap in your productivity and programming throughput. However, you soon came to see that although the 1541 allows you to harness some of the power of the Commodore 64, it can be cumbersome to use. Many programmers have set out to help rectify this by designing various utilities that make using a disk drive easier. While these solve some of the difficulties of disk drive usage, they create another problem: you end up with a disk full of programs, one to do single-file copies, another to list the directory, another that allows you to 'unscratch' a file, and still another to find the load address of a file. Whew! Are things really easier now?

I thought not—so I sat down, and after a little head scratching came up with *Dos Plus. Dos Plus* gives you a powerful set of tools with which you can better manage your 1541 disk drive. The real power of *Dos Plus* is that you needn't keep an arsenal of utilities on disk waiting to be used. *Dos Plus* is in itself just such an arsenal. You need only load it once. It will tuck itself away above BASIC memory and wait patiently for your commands.

First type in *DOS Boot* (see page 98) and save it to disk. Then, using the *Flankspeed* program on page 86, type in and save *DOS Plus*. Then type NEW, and load and run *DOS Boot*.

DOS Boot will also let you change certain default parameters of your copy of *Dos Plus*. More on that later in this article.

You now have an error-free working copy of *Dos Plus*. But what exactly do you have? Let's see. First, list the directory by typing '@$'. *Dos Plus* gives much useful information here. The load address of the program is listed to the right of your screen in hex and decimal. Block count is given, as in other Wedge programs. If your disk contains deleted files they will be listed as such, with a 'DEL' signifying the fact. You may abort the listing of a directory by holding down the RUN/STOP key. You may have to hold this key until it 'catches,' because the keyboard is polled between accesses to disk. Likewise, the space bar will pause a listing, with any other key cancelling the pause function. Again, you will have to hold the space bar until the pause takes effect. If you wish to list the directory directly to a printer on the serial port, hit the f1 or f3 keys before you issue the @$ command. If your printer answers to device number 4, use the f1 key. If your printer answers to device number 5, use the f3 key. Be careful that you use the correct key for your printer. If the computer attempts to open a channel to a non-existent device, the program will 'hang.' You can recover from this only by using the RUN/STOP-RESTORE combination. If you accidentally press f1 or f3 or wish to cancel a dump to the printer, POKE 923,0 to cancel the dump flag. The dump to printer function is automatically cancelled after the dump is completed or, alternately, if you abort the total listing of the directory with the RUN/STOP key.

Let's try another tool. Type the following:

```
@:D18,1 <RETURN>
```

This is the SECTOR DUMP/MODIFY command. Magically, the first sector of the disk directory is listed to the screen. You may use the cursor keys to move the cursor to any position within the block. If you make any changes to the block, the entire block will be rewritten to disk as modified. If you accidentally enter an invalid digit, i.e., 0Z or P8, the rewrite feature will abort. You can use this command structure to do many things. You can unscratch a file by changing the third byte preceding the first character of a filename from 00 to 82. The unscratch will be successful only if no saves were made to disk after the file was scratched. This feature is primarily useful when you inadvertently scratch a file. Alternately, you might wish to change a sequential file to a program file or a user file to a sequential file. You should not make changes concerning relative files. A sector, or block, can be dumped directly to your printer in the same manner as described for directory listings to printer. Again, make use of the f1/f3 keys prior to issuing the @:D command. After the printer dump is complete, the sector will be listed to the screen where you may modify it or exit by pressing RETURN. I would advise that you practice using the @:D function on an expendable disk. This function is very powerful and can generate disastrous results if used carelessly.

This one you've already used:

```
@:[BACKARROW]'FILENAME',$<START ADDRESS>
,$<END ADDRESS+1>
```

You use this command structure to save a file from other

Continued on page 61

By Dennis Muscatelli

COMMODORE 64

* with $19.95 Software Purchase

$139.00*

- 170K Disk Drive $149.00 *
- Tractor Friction Printer $169.00
- 13" Hi-Res Color Monitor $189.00 *

* See Page 13

COMPUTER AND SOFTWARE SALE

CALL BEFORE YOU ORDER **PRICES MAY BE LOWER**

SUPER AUTO DIAL MODEM COM 64

(Best communications package in USA)

$59.00

- Computer Learning Pad $37.95
- New Voice Synthesizer $49.00
- 12" Green or Amber Monitor $79.95
- 12" Daisy Wheel Printer $199.00

*** COMMODORE 64 COMPUTER $139.00**

You pay only $139.00 (with the $19.95 software purchase, see below) when you order the powerful 84K COMMODORE 64 COMPUTER! LESS the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your computer that allows you to SAVE OVER $500 off software sale prices!! With only $100 of savings applied, your net computer cost is $39.00!!

*** 170K DISK DRIVE $149.00**

You pay only $149.00 (with the $19.95 software purchase, see below) when you order the 170K Disk Drive! LESS the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your disk drive that allows you to SAVE OVER $500 off software sale prices!! With only $100 of savings applied, your net disk drive cost is $49.00.

*** 13" HI-RES COLOR MONITOR $189.00**

You pay only $189.00 (with the $19.95 software purchase, see below) when you order this 13" COLOR MONITOR with sharper and clearer resolution than any other color monitors we have tested! LESS value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your monitor that allows you to save over $500 off software sale prices! With only $100 of savings applied, your net color monitor cost is only $89.00. (16 Colors).

80 COLUMN 80 CPS TRACTION/FRICTION PRINTER $169.00

You pay only $169.00 when you order the Comstar T/F deluxe line printer that prints 8½x11 full size, single sheet, roll or fan fold paper, labels, etc. Impact dot matrix, bidirectional. LESS the value of the SPECIAL SOFTWARE DISCOUNT COUPON we pack with your printer that allows you to SAVE OVER $500 off software sale prices!! With only $100 of savings applied your net printer cost is only $69.00.

80 COLUMN BOARD $79.00

Now you program 80 COLUMNS on the screen at one time! Converts your Commodore 64 to 80 COLUMNS when you plug in the 80 COLUMN EXPANSION BOARD!! PLUS 4 slot expander! Can use with most software.

80 COLUMNS IN COLOR EXECUTIVE WORD PROCESSOR $39.00

This EXECUTIVE WORD PROCESSOR is the finest available for the COMMODORE 64 computer! The ULTIMATE FOR PROFESSIONAL Word Processing, DISPLAYS 40 or 80 COLUMNS IN COLOR or black and white! Simple to operate, powerful text editing with 250 WORD DICTIONARY, complete cursor and insert/delete key controls line and paragraph insertion, automatic deletion, centering, margin settings and output to all printers! Includes a powerful mail merge.
List $99.00 **SALE $39.00** Coupon $29.95

SPECIAL SOFTWARE COUPON

We pack a SPECIAL SOFTWARE DISKCOUNT COUPON with every COMMODORE 64 COMPUTER, DISK DRIVE, PRINTER, or MONITOR we sell! This coupon allows you to SAVE OVER $500 OFF SALE PRICES!!

(Examples)

PROFESSIONAL SOFTWARE COMMODORE 64

Name	List	Sale	Coupon
Executive Word Processor	$99.00	$39.00	$29.95
Executive Data Base	$69.00	$29.00	$19.95
20,000 Word Dictionary	$24.95	$14.95	$10.00
Electronic Spread Sheet	$59.95	$49.00	$39.00
Accounting Pack	$49.00	$39.00	$29.00
Practicalc	$59.95	$34.95	$29.95
Programmers Reference Guide	$20.95	$16.95	$12.50
Programmers Helper (Disk)	$59.95	$29.95	$19.95
80 Column Screen (Disk)	$59.95	$29.95	$19.95
Disk Filer (by Flip-N-File)	$39.95	$14.95	$12.95
Deluxe Tape Cassette	$89.00	$44.95	$34.95
Pro Joy Stick	$19.95	$12.95	$10.00
Light Pen	$39.95	$14.95	$ 9.95
Dust Cover	$ 8.95	$ 6.95	$ 4.60
Simon's Basic	$29.95	$22.95	$19.95
Pitstop II Epyx	$39.95	$24.95	$19.95
Super Graphics Expander	$29.95	$22.95	$19.95
Music Calc 1	$59.95	$29.95	$24.95
Filewriter	$59.95	$29.95	$24.95

(See over 100 coupon items in our catalog)

Write or call for Sample SPECIAL SOFTWARE COUPON!

EXECUTIVE QUALITY PROFESSIONAL BUSINESS SOFTWARE

The Cadillac of Business Programs for Commodore 64 Computers

Item	List	* Sale	Coupon
Inventory Management	$79.95	$29.95	$24.95
Accounts Receivable	$79.95	$29.95	$24.95
Accounts Payable	$79.95	$29.95	$24.95
Payroll	$79.95	$29.95	$24.95
General Ledger	$79.95	$29.95	$24.95

The $19.95 Software Purchase Options

	LIST	SALE
1. 24 Program Bonus Pack (tape or disk)	$29.95	**$19.95**
2. Oil Barrons-Strategy Board Game	$49.95	**$19.95**
3. Disk Drive Cleaner	$29.95	**$19.95**
4. HES Games (disk)	$29.95	**$19.95**
5. Pogo Joe (tape or disk)	$29.95	**$19.95**

SUPER AUTO DIAL MODEM $59.00

Easy to use. Just plug into your Commodore 64 computer and you're ready to transmit and receive messages. Easier to use than dialing your telephone, just push one key on your computer! Includes exclusive easy to use program for up and down loading to printer and disk drives.
List $129.00 **SALE $59.00**

NEW COMPUTER LEARNING PAD $37.95

Makes other graphics tablets obsolete. This new TECH SKETCH LEARNING PAD allows you to draw on your T.V. or Monitor and then you can print whatever you draw on the screen on your printers. FANTASTIC!!!
List $79.95 **SALE $37.95**

NEW VOICE SYNTHESIZER $49.00

For Com-64 or VIC-20 computers. Just plug it in and you can program words and sentences, adjust volume and pitch, make talking adventure games, sound action games and customized talkies!! FOR ONLY $19.95 you can add TEXT TO SPEECH, just type a word and hear your computer talk — ADD SOUND TO "Zork", SCOTT ADAMS AND AARDVARK ADVENTURE GAMES!! (Disk or tape.)

12" GREEN OR AMBER MONITOR $79.95

Your choice of green or amber screen monitor, top quality, 80 columns x 24 lines, easy to read, anti-glare! PLUS $9.95 for connecting cable. Com-64 or VIC-20.

12" DAISY WHEEL PRINTER $199.00

"JUKI" Superb letter quality daisy wheel printer, 12" extra large carriage, up to 12 CPS bi-directional printing, drop in cassette ribbon, centronics parallel or RS232 serial port built in! (Specify)
List $299.00 **SALE $199.00**

CARDCO G+ INTERFACE $59.00

For Commodore 64 and Vic 20 computers. Lets you use other printers with Centronics interfaces. This interface lets the printer act like a Commodore printer including printing the Commodore graphics (Dot matrix with graphic capability printers).
List $109.00 **SALE $59.00**

PROTECTO WARRANTY

All Protecto's products carry a minimum 90 day warranty. Therefore, if anything fails within 90 days from the date of purchase, you simply send your product to us via United Parcel Service prepaid. We will IMMEDIATELY send you a replacement at no charge via United Parcel Service prepaid. This warranty proves once again that ***We Love Our Customers.***

• LOWEST PRICES • 15 DAY FREE TRIAL
• BEST SERVICE IN U.S.A. • ONE DAY EXPRESS MAIL

PHONE ORDERS
8 a.m. - 8 p.m. Weekdays
9 a.m. - 12 noon Saturdays

• 90 DAY FREE REPLACEMENT WARRANTY
• OVER 500 PROGRAMS • FREE CATALOGS

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D. No C.O.D. to Canada, APO-FPO

PROTECTO

We Love Our Customers

Box 550, Barrington, Illinois 60010

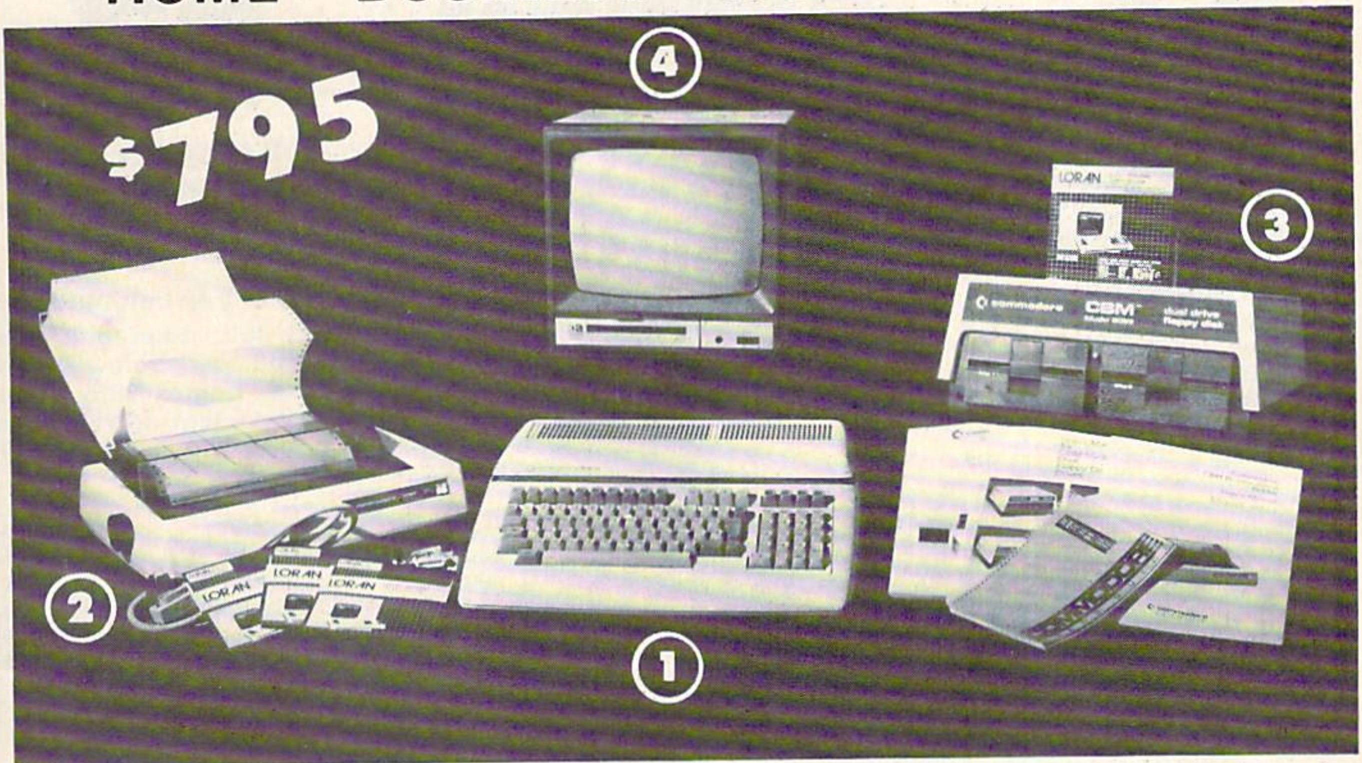
312/382-5244 to order

Reader Service No. 221

NEW 128K —MEGA BYTE DUAL DISK DRIVE—80 COLUMN

COMPUTER SYSTEM SALE!

HOME • BUSINESS • WORD PROCESSING



LOOK AT ALL YOU GET FOR ONLY **$795.**

	LIST PRICE
① B128 COMMODORE 128K 80 COLUMN COMPUTER	$ 995.00
② 4023 - 100 CPS - 80 COLUMN BIDIRECTIONAL PRINTER	499.00
③ 8050 DUAL DISK DRIVE (over 1 million bytes)	1795.00
④ 12" HI RESOLUTION 80 COLUMN MONITOR	249.00
• BOX OF 10 LORAN LIFETIME GUARANTEED DISKS	49.95
• 1100 SHEETS FANFOLD PAPER	19.95
• ALL CABLES NEEDED FOR INTERFACING	102.05
TOTAL LIST PRICE	**$3717.95**

$795 SALE PRICE

PLUS YOU CAN ORDER THESE BUSINESS PROGRAMS AT SALE PRICES

Programmers Reference Guide
List $29.95
Sale $24.95

	LIST	SALE	SYSTEM PRICE*
Professional 80 Column Word Processor	$149.95	**$99.00**	***$49.00***
Professional Data Base	$149.95	**$99.00**	***$49.00***
Accounts Receivable	$149.95	**$99.00**	
Accounts Payable	$149.95	**$99.00**	

	LIST	SALE
Payroll	$149.95	**$99.00**
Inventory	$149.95	**$99.00**
General Ledger	$149.95	**$99.00**
Financial Spread Sheet	$149.95	**$99.00**
Order Entry	$149.95	**$99.00**

* When Bought with B128 Computer System.

☆ PRINTER REPLACEMENT OPTIONS
(replace the 4023 with the following printers at these sale prices. Interface Included)

	LIST	SALE
☆ Olympia Executive Letter Quality Printer	$699.00	**$369.00**
☆ Comstar Hi-Speed 160 CPS 15½" Business Printer	$779.00	**$459.00**
Telecommunications Deluxe Modem Package	$199.00	**$139.00**
IEEE to Centronics Parallel Printer Interface	$179.00	**$139.00**

15 DAY FREE TRIAL. We give you 15 days to try out this SUPER SYSTEM PACKAGE!! If it doesn't meet your expectations, just send it back to us prepaid and we will refund your purchase price!!
90 DAY IMMEDIATE REPLACEMENT WARRANTY. If any of the SUPER SYSTEM PACKAGE equipment or programs fail due to faulty workmanship or material we will replace it IMMEDIATELY with no service charge!!

Add $50.00 for shipping and handling!!
WE DO NOT SHIP THIS SYSTEM TO FOREIGN COUNTRIES, CANADA, PUERTO RICA, ALASKA, OR HAWAII.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! We accept Visa and MasterCard. We ship C.O.D. to continental U.S. addresses only.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

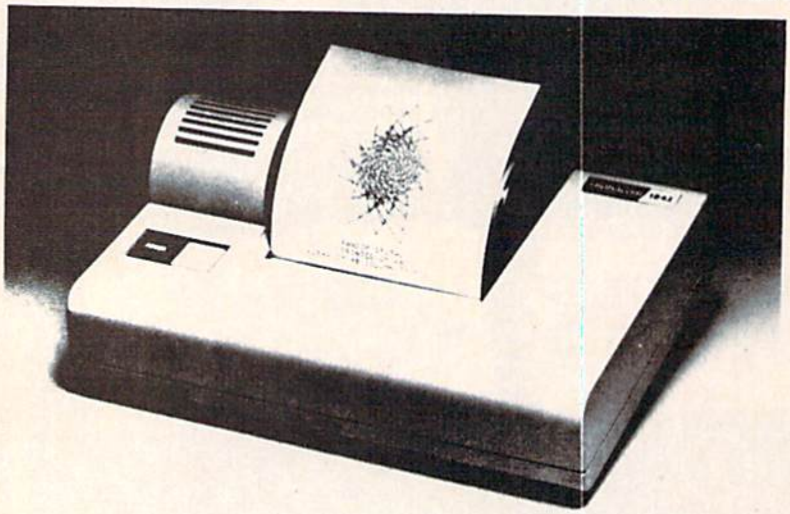
Reader Service No. 222

FANTASTIC PRINTER

$24.95 SALE!! $24.95

Commodore 64 — VIC 20 — Atari

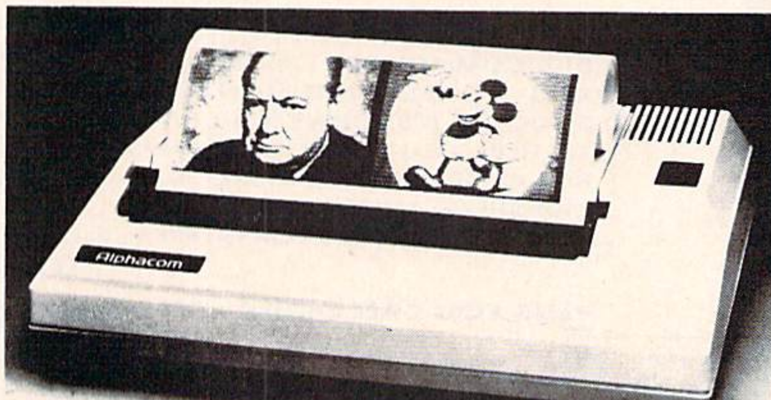
☆ 40 And 80 Column Printers ☆ Up To 100 Characters Per Second ☆ Full Graphics Capability ☆ Upper And Lower Case ☆ Advanced Thermal Technology For Quiet Operation



3100 Alphacom 42-80 CPS 40 Column Printer
Print out listings with full computer character sets (interface required, see below). Print in upper and lower case. Comes with a roll of paper and all power adapters needed. Perfect for a spare printer or program lister. List $99.00. **Sale $24.95.**

40 Column Extra Paper	LIST	SALE
3103 25 Meter Rolls Blue	$9.95	**$3.00**
3104 40 Meter Rolls Blue	$16.95	**$3.95**
3105 25 Meter Rolls Black (1 per pkg.)	$12.95	**$3.95**
3106 25 Meter Rolls Blue (2 per pkg.)	$19.95	**$5.95**
3107 25 Meter Rolls Black (2 per pkg.)	$19.95	**$5.95**

3150 Alphacom 81-100 CPS 80 Column Printer
Now you can have a printer for the cost of a large box of paper. This printer prints in upper and lower case with true lower descenders. Comes with 1 roll of paper and power adapter. With the intelligent interfaces (sold below) you can do Ascii graphics as well as Atari or Commodore graphics. Plus you can do underlining and expanded modes. Print out pictures, program listings, word processing pages, etc. Perfect for the student or homeowner. List $199.00. **Sale $39.95.**



80 Column Extra Paper	LIST	SALE
3153 40 Meter Rolls Blue	$14.95	**$3.95**
3154 40 Meter Rolls Black	$19.95	**$4.95**
3155 25 Meter Rolls Blue (2 per pkg.)	$19.95	**$7.95**
3156 25 Meter Rolls Black (2 per pkg.)	$19.95	**$8.95**

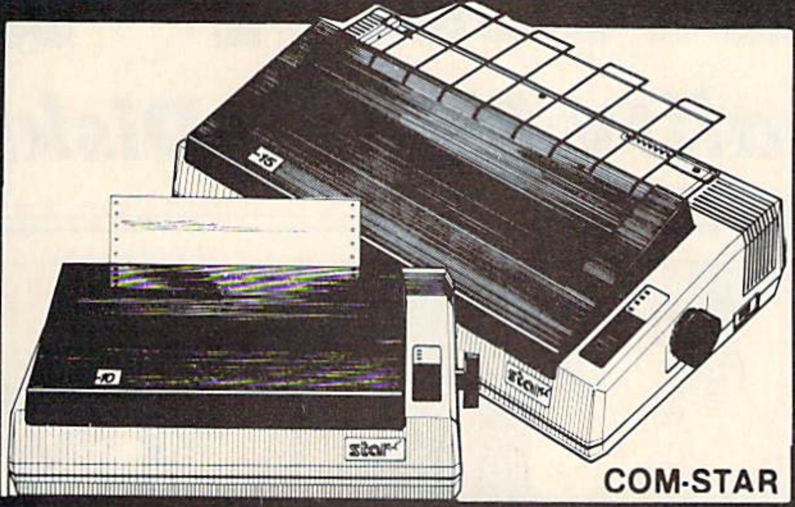
3101 Intelligent Commodore Interface — Allows you to hook the 40 or 80 column printer to the Commodore 64, do program listings, allows software screen dumps, etc. Includes Commodore graphics and reverse characters. (Specify 40 or 80 Column) List $59.95.
40 Column Sale $9.95. **80 Column Sale $14.95.**

3102 Intelligent Atari Interface — Allows you to hook the 40 or 80 column printer to the Atari computer, do program listings, allows software screen dumps, etc. Includes Atari graphics and reverse characters. (Specify 40 or 80 Column) List $59.95.
40 Column Sale $9.95. **80 Column Sale $14.95.**

Add $10.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $20.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES EXCEPT CANADA.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D. No C.O.D. to Canada, APO-FPO

PROTECTO
We Love Our Customers
Box 550, Barrington, Illinois 60010
312/382-5244 to order

FANTASTIC COMPUTER PRINTER SALE!!!



COM-STAR T/F

Tractor Friction Printer

only $ 169**

- Lowest Priced, Best Quality, Tractor-Friction Printers in the U.S.A.
- Fast 80-120-160 Characters Per Second
- 40, 46, 66, 80, 96, 132 Characters Per Line Spacing
- Word Processing
- Print Labels, Letters, Graphs and Tables
- List Your Programs
- Print Out Data from Modem Services
- "The Most Important Accessory for Your Computer"

****DELUXE COMSTAR T/F 80 CPS Printer $169.00**

This COMSTAR T/F (Tractor Friction) PRINTER is exceptionally versatile. It prints 8½" x 11" standard size single sheet stationery or continuous feed computer paper. Bi-directional, impact dot matrix, 80 CPS, 224 characters. (Centronics Parallel Interface).

Premium Quality 120-140 CPS 10X COM-STAR PLUS+ Printer $199

The COM-STAR PLUS+ gives you all the features of the COMSTAR T/F PRINTER plus a 10" carriage, 120-140 CPS, 9 x 9 dot matrix with double strike capability for 18 x 18 dot matrix (near letter quality), high resolution bit image (120 x 144 dot matrix), underlining, back spacing, true lower descenders with super & subscripts, prints standard, italic, block graphics & special characters. It gives you print quality and features found on printers costing twice as much!! (Centronics Parallel Interface) (Better than Epson FX80). RATED "BEST BUY" BY CONSUMERS DIGEST BUYERS GUIDE, JANUARY 1985! List $499.00. Sale $199.00.

Premium Quality 120-140 CPS 15½X COM-STAR PLUS+ Business Printer $319.00

Has all the features of the 10" COM-STAR PLUS+ PRINTER plus 15½" carriage and more powerful electronics components to handle large ledger business forms! (Better than Epson FX 100). List $599. Sale $319.00.

Superior Quality 15½" COM-STAR PLUS+ H.S. High Speed 160-180 CPS Business Printer $449.00

15½" Carriage and more powerful electronic components to handle larger ledger business forms! (Serial Centronics Parallel Interface). List $799. Sale $449.

12" DAISY WHEEL PRINTER $199.00

"JUKI" Superb letter quality daisy wheel printer, 12" extra large carriage, up to 12CPS bi-directional printing, drop-in cassette ribbon, (90 day warranty) centronics parallel or RS232 serial port built in! (Specify). List $299.00. Sale $199.00.

Superior Quality 10" COM-STAR+ H.S. HIGH SPEED 160-180 CPS Business Printer $349.00

This **Super High Speed Com-Star+ Business Printer** has all the features of the 10" COM-STAR+ PRINTER with HIGH SPEED BUSINESS PRINTING 160-180 CPS, 100% duty cycle, 8K Buffer, diverse character fonts, special symbols and true descenders, vertical and horizontal tabs. **A RED HOT BUSINESS PRINTER** at an unbelievable low price (Serial or Centronics Parallel Interface). List $699.00. Sale $349.00.

Printer/Typewriter Combination $249.00

"JUKI" Superb letter quality, daisy wheel printer/typewriter combination. Two machines in one — just a flick of the switch. 12" extra large carriage, typewriter keyboard, automatic margin control and relocate key drop in cassette ribbon! (90 day warranty) centronics parallel or RS232 serial port built in (Specify). List $399.00. $249.00.

• 15 Day Free Trial — 1 Year Immediate Replacement Warranty

PARALLEL INTERFACES

For VIC-20 and COM-64 — $49.00. Commodore B-128 — $139.00. Apple — $79.00. Atari — $79.00.

Add $14.50 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $29.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!

VISA—MASTER CARD—We Ship C.O.D. to U.S. Addresses Only

PROTECTO

We Love Our Customers

Box 550, Barrington, Illinois 60010

312/382-5244 to order

COM-STAR PLUS+ Print Example: ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ 1234567890

Reader Service No. 224

FUJI SALE $1.49 ea.

Premium Quality Floppy Disks



Box of 10 *with hub rings, sleeves and labels*

Famous Brand FUJI Floppy Disks for those who care about keeping their data.

Single Sided — Double Density

for Commodore 64, Atari, Apple

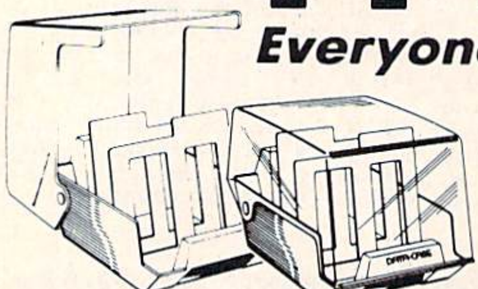
List $39.95

Sale $14.90

*** $12.95** ☆ ☆ ***Flip-N-File*** ☆ ☆ *** $12.95**

Data-Case

Floppy Disk Filer



Everyone Needs a Floppy Disk Secretary

Facts:

- Dust and Dirt particles can hurt your disks
- Most disks go bad due to mishandling in storage
- Proper filing of your disk collection will reduce unnecessary handling of your disks

The Floppy Disk Filer is an inexpensive hard plastic Fliptop case that will allow for easy filing, and protect your disks from dust, smoke, and dirt. Plus, the Floppy Disk Filer will keep all your disks out of unwanted hands and in one place where you can easily find them. **(Holds Over 50 Disks)**

List $24.95 **Introductory Sale Price $14.95** **** Coupon $12.95***

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!

VISA — MASTER CARD — C.O.D.

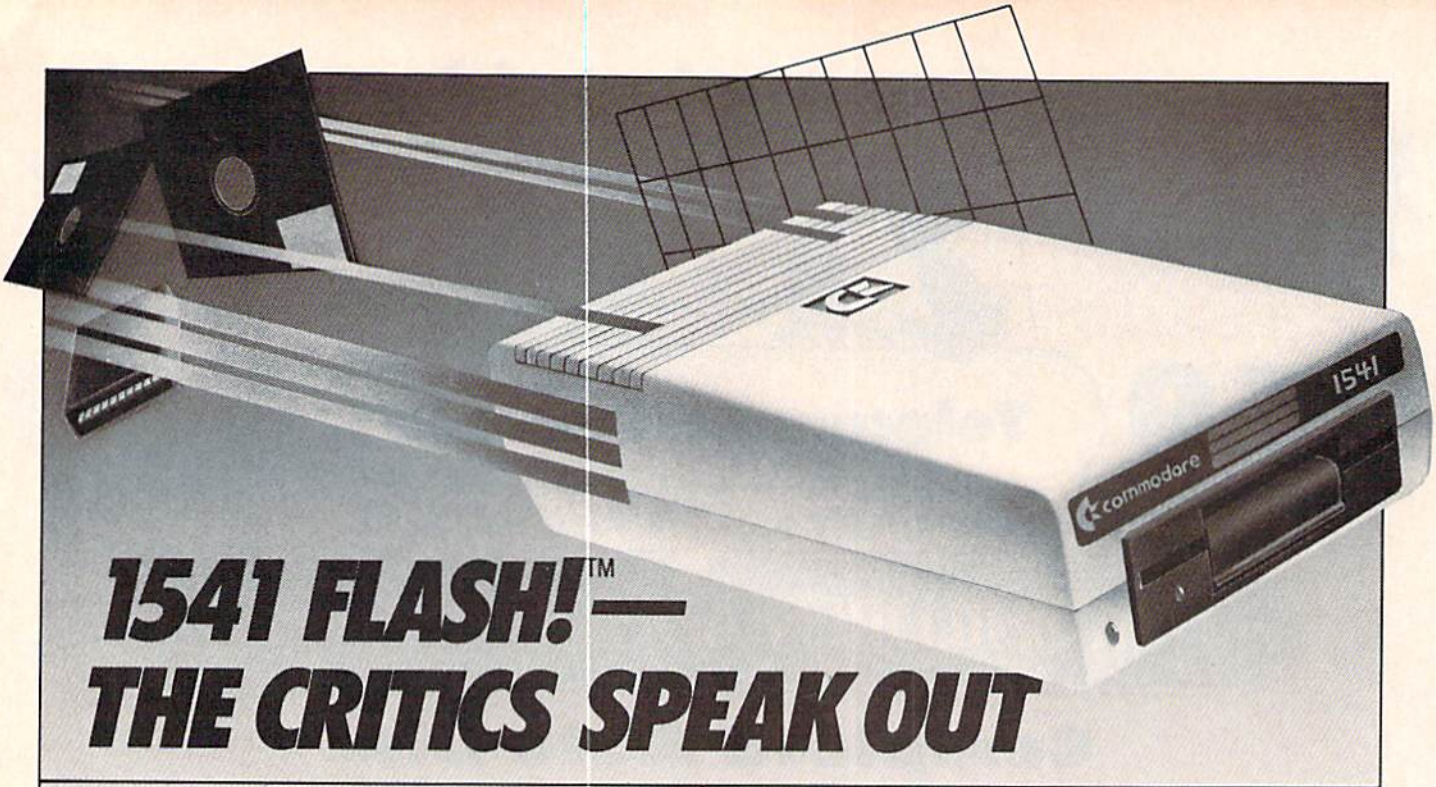
No C.O.D. to Canada, APO-FPO.

PROTECTO

We Love Our Customers

Box 550, Barrington, Illinois 60010

312/382-5244 to order



1541 FLASH!™— THE CRITICS SPEAK OUT

This advertisement has been written by independent reviewers of Commodore computer products. We thank them for their frankness and the high marks awarded to the Skyles Electric Works **1541 FLASH!**.

The Reviewers Comment

"Tired of that slow disk drive? The solution is here, and it costs about $90. Considering what it does, the price is a bargain! It speeds up your 1541 disk drive 200%–300%. And if you write your own software specially for the **FLASH!**, you can achieve speed increases of 600%!

The **1541 FLASH!** is the best! It's better than KWIK LOAD.™ And better than 1541 Express.™

Does it sound too good to be true? Do you suspect there must be some drawback that I haven't mentioned? Well... There is one. You have to open up your keyboard and 1541 drive and do a little work inside them. You need to replace a couple of chips with new ones provided by the **FLASH!**. And an extra cable will run from your keyboard's user port to your 1541 drive. But the installation is explained in complete detail with pictures. It's a simple operation that will take under 30 minutes. And in return you will have a disk drive that literally races along!

The biggest complaint with the Commodore 64 is the slow disk drive. No more! You will never be willing to go back after having used the **1541 FLASH!**. It really spoils you! It's even faster than a parallel drive with an IEEE interface! Don't be afraid of the installation. It's really simple. And if you prefer not to do it yourself, your local user's group probably has people with the ability to install it for you. You'll be glad you did!"

The Northwest Users Guide, Jan. 1985

"A tiny wedge is included... you simply SYS65526 to enable it. Those who enjoy using the wedge as part of their normal computer will like this feature.

The utilities added by **FLASH!** include single, double and simultaneous keystroke implementation of such niceties as delete line, escape quote, cursor to bottom of screen, 16 character tab, and return without line execution.

My children have played some of their favorite games and used utilities or educational software without any problems whatsoever, using the **1541 FLASH!** All in all, this device will save the purchaser much of the frustration normally experienced, as well as the time required in normal disk drive-computer interactions. With the above noted exception, I'm very pleased with its operation, and won't hesitate to recommend it to those who would like faster loads and saves or want additional flexibility and power at a moderate price."

RUN, May 1985

"Having used **1541 FLASH!** for several days, already I can't bear to go back to a slow 1541. It's amazing how quickly you can get spoiled by a luxury like this. More than a few editors here have cast covetous eyes on the upgraded 64/1541. And the price is reasonable for such a dramatic enhancement. After reading the installation description above, you can decide if you're up to opening your equipment, yanking out chips, and enjoying a 300% speedup."

Compute!'s Gazette, Jan. 1985

Commercial Details

1541 FLASH!	**$ 89.95***
1541 FLASH! with Disk Switchboard	**99.00***
1541 FLASH! for Two Drives	**139.95***
1541 FLASH! for SX-64	**99.95***
1541 FLASH! for MSD Drives	**tba***

*Mail order please add $3.50 for shipping and handling, California residents add sales tax as required.

Available from your local Commodore dealer or call 1-800-227-9998

Skyles Electric Works
231E South Whisman Road
Mountain View, CA 94041
1-(415) 965-1735

1541 FLASH! is a trademark of Skyles Electric Works.
Commodore 64 is a trademark of Commodore.

COLUMN 2: By the Numbers

The Binary, Hexadecimal, and Decimal Notation Systems

By Mark Andrews

Three kinds of number systems are commonly used in assembly language programming. They are:

(1) *Decimal numbers,* based on the value 10 and written using the Arabic numerals 0 through 9.

(2) *Binary numbers,* based on the value 2 and usualy written using the Arabic numerals 0 and 1.

(3) *Hexadecimal numbers,* based on the value 16 and customarily written using the Arabic numbers 0 through 9, plus the letters A through F. The digits 1 through 9 are used just as in conventional decimal notation, and the letters A through F as single-unit symbols for the values 10 through 15.

Decimal numbers are what most of us learned in school so not much will be said about them. Instead, we'll focus on the binary and hexadecimal number systems. Let's start with binary numbers.

The ones and zeroes that make up binary numbers are known as *bits.* A series of four bits is called a *nibble* (or *nybble*), a series of eight bits is called a *byte*, and a series of 16 bits is called a *word.*

PENGUIN MATH

The best way to explain the principles of the binary system is with something I call Penguin Math. Penguin Math is the number system that penguins would probably use if penguins could use numbers.

Imagine that you're a penguin. You wouldn't have 10 fingers on each hand, so there'd be no way you could count on your fingers. You'd have just two flippers, and if you wanted to count on something, you'd have to count on them.

Suppose that you're a very bright penguin, and did learn to count on your flippers. You wouldn't be able to count to 10. But you would be able to count to 2.

Now suppose that you're the smartest penguin on Penguin Island. If you were that sharp, you might figure out how to use your flippers to count past 2. Here's one way you could do that.

Instead of counting on your flippers the same way humans count on their fingers, you could decide to equate a raised right flipper to 1, and a raised left flipper to 2. Then you could let two raised flippers represent the value 3.

Now suppose you were the Albert Einstein of Penguin Island, and devised a new mathematical system to express in writing what you had done.

You could use a 0 to represent an unraised flipper, and a 1 to represent a raised one. And then you could scratch these equations in the ice:

00 = 0
01 = 1
10 = 2
11 = 3

AHA! FEET, TOO!

Those are, of course, binary numbers. And they clearly show that if you were a really smart penguin, you could use two flippers to express the values 0 through 3. That's a clear improvement over the two values that can be represented with more traditional flipper-counting methods.

Since our imagination has taken us this far, let's now suppose that you (still a penguin) wanted to learn to count past 3. While pondering this problem, you might look down at your feet–and notice two more flippers down there.

Voila–bigger numbers!

By using both flippers and both legs at the same time, you could count as follows:

FOUR-BIT PENGUIN MATH

0000 = 0
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9

...and so on.

If you continued counting like this, you would eventually discover that you could express 16 values—0 through decimal 15—using four-digit (or four-bit) binary numbers.

ONE MORE LESSON

Now we're ready for one last lesson in Penguin Math. Imagine that you marry another penguin. And, using your skill with binary numbers, you determine that you and your spouse have a total of eight flippers between you.

If your spouse decides to cooperate with you in counting with flippers, the two of you could now sit on your ice and start a floe chart with numbers that look like these:

0000 0001 = 1
0000 0010 = 2
0000 0011 = 3
0000 0100 = 4
0000 0101 = 5

If you and your spouse kept on counting in this fashion—using 8-bit Penguin Math—you would eventually discover that by using eight flippers, you could count from 0 (or 0000 0000) to 255 (or 1111 1111), for a total of 256 values.

That completes this crash course in Penguin Math. What it has demonstrated is that it is possible to express 256 values—from 0000 0000 through 1111 1111—using 8-bit binary numbers.

THE HEXADECIMAL NUMBER SYSTEM

Since computers "think" in binary numbers, the binary system is obviously an excellent notation system for representing computer data. But, as you may remember from last month's column, binary numbers have one serious shortcoming: they're extremely difficult to read. So the binary system is not the numeric system most often used in assembly language programming. The numeric system you'll encounter most often in assembly language programming is *hexadecimal.*

Just as binary numbers are based on the value 2, hexadecimal numbers are based on the value 16. If people were born with eight fingers on each hand, we would probably all count in hexadecimal numbers.

Hexadecimal numbers are often used in assembly language programming because they can help bridge the gap between the binary and decimal systems. Since binary numbers have a base of 2, and hex numbers have a base of 16, a series of four binary bits can always be translated into one hexadecimal digit. So a series of eight bits (a "byte") can always be represented by a pair of hexadecimal digits, and a series of 16 bits (a "word") can always be represented by a four-digit hexadecimal number.

Here is a table comparing decimal, hexadecimal, and binary numbers. Observe that odd-looking letter-and-number combinations like "FC1C", "5DA4", and even "ABCD" are perfectly good numbers in the hexadecimal system.

DECIMAL	HEXADECIMAL	BINARY
1	1	00000001
2	2	00000010
3	3	00000011
4	4	00000100
5	5	00000101
6	6	00000110
7	7	00000111
8	8	00001000
9	9	00001001
10	A	00001010
11	B	00001011
12	C	00001100
13	D	00001101
14	E	00001110
15	F	00001111
16	10	00010000

As you can see, the decimal number 16 is written "10" in hex and "00010000" in binary, and is thus a round number in both the binary system and the hexadecimal system. And the hexadecimal digit F, which comes just before hex 10 (or 16 in decimal), is written 00001111 in binary.

As you become more familiar with the binary and hexadecimal systems, you will begin to notice many other similarities. For example, the decimal number 255 (the largest 8-bit number) is 11111111 in binary and FF in hex. The decimal number 65,535 (the highest memory address in a 64K computer) is written FFFFFFFF in hex and 11111111 11111111 in hex.

CONVERTING BINARY NUMBERS TO DECIMAL NUMBERS

It isn't very difficult to convert a binary number to a decimal number. In a binary number, as we've seen, the bit farthest to the right represents 2 to the power 0. The next bit to the left represents 2 to the power 1, the next represents 2 to the power 2, and so on.

The digits in an 8-bit binary number are therefore numbered 0 to 7, starting from the rightmost digit. The rightmost bit—often referred to as Bit 0—represents 2 to the 0th power, or the number 1. And the leftmost bit—often called Bit 7—is equal to 2 to the 7th power, or 128.

The following is a list of simple equations that illustrate what each bit in an 8-bit binary number means.

VALUES OF THE BITS IN AN 8-BIT BINARY NUMBER

Bit 0 = 2 to the 0th power =	1
Bit 1 = 2 to the 1st power =	2
Bit 2 = 2 to the 2nd power =	4
Bit 3 = 2 to the 3rd power =	8

Bit 4	= 2 to the 4th power	=	16
Bit 5	= 2 to the 5th power	=	32
Bit 6	= 2 to the 6th power	=	64
Bit 7	= 2 to the 7th power	=	128

The above table provides an easy method of converting any 8-bit binary number into its decimal equivalent. Instead of writing the number down from left to right, write it instead in a vertical column, with Bit 0 at the top and Bit 7 at the bottom. Then multiply each bit in the binary number by the decimal number that it represents. Then add up the results of all of these multiplications—and the total you get will be the decimal value of the binary number.

Suppose, for example, that you wanted to convert a binary number 00101001 into a decimal number. Here's how you could do it:

CONVERTING A BINARY NUMBER INTO A DECIMAL NUMBER

1 ×	1	=	1
0 ×	2	=	0
0 ×	4	=	0
1 ×	8	=	8
0 ×	16	=	0
1 ×	32	=	32
0 ×	64	=	0
0 ×	128	=	0
TOTAL		=	41

00101001 AND OTHER FANCY NUMBERS

If the calculation in the above example is correct, the binary number 00101001 should be equivalent to the decimal number 41 (that's decimal 41, not hex $41, for those already familiar with hexadecimal notation). Look up either 00101001 or 41 on any binary-to-decimal or decimal-to-binary conversion chart, and you'll see that the calculation was accurate. And this conversion technique will work with any other binary number.

CONVERTING A BINARY NUMBER TO A DECIMAL NUMBER

Now we'll go in the other direction, and convert a decimal number to a binary number.

First, divide the number by 2. Then write down both the quotient and the remainder. Since we're dividing by 2, the remainder will be either a 1 or 0. So what we write down will be the quotient followed by either a 1 or a 0.

Next we'll take the quotient, divide it by two, and write the result down. If there's a remainder (a 1 or a 0), we'll write that down, too, underneath the first remainder.

When there are no numbers left to divide, we'll write down all of the remainders we got, reading from the bottom to the top. What we'll have then, of course, is a binary number—a number made up of ones and zeroes. And that number will be the binary equivalent of the decimal number we started out with.

This conversion technique is illustrated in the following example:

CONVERTING A DECIMAL NUMBER TO A BINARY NUMBER

117/2	=	58 with a remainder of 1
58/2	=	29 with a remainder of 0
29/2	=	14 with a remainder of 1
14/2	=	7 with a remainder of 0
7/2	=	3 with a remainder of 1
3/2	=	1 with a remainder of 1
1/2	=	0 with a remainder of 1
0/2	=	0 with a remainder of 0

HOW TO DO IT

To complete the decimal-to-binary conversion presented in this example, simply copy the binary digits in the right-hand column, writing them down horizontally from right to left, with the top digit on the right. You'll then see that the binary equivalent of the decimal (not hexadecimal) number 117 is 01110101. If you have a decimal-to-binary conversion chart handy, you can use it to confirm this.

BINARY-TO-HEX AND HEX-TO-BINARY CONVERSIONS

It's easy to convert binary numbers to their decimal equivalents. Just use this chart:

HEXADECIMAL-TO-BINARY CONVERSION CHART

HEXADECIMAL	BINARY
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

The above table shows how to convert a multiple-digit hex number to binary notation: merely string the hex digits together and convert each one separately. For example, the binary equivalent of the hexadecimal number C0 is 1100 0000. The binary equivalent of the hex number 8F2 is 1000 1111 0010. The binary equivalent of the hex number 7A1B is 0111 1010 0001 1011.

To convert binary numbers to hexadecimal numbers, use the chart in reverse. The binary number 1101 0110 1110 0101, for example, is equivalent to the hexadecimal number D6E5.

DECIMAL-TO-HEXADECIMAL CONVERSION

It's almost as easy to convert decimal numbers to hexadecimal as it is to translate binary to decimal. Here's how it's done:

First, take a decimal integer that you want to convert, and divide it by 16. Then write down the remainder, like this:

64540/16 = 4033 with a remainder of 12

Then divide the integer part of the above quotient by 16, and write down the result of that calculation:

4033/16 = 252 with a remainder of 1

Keep repeating this process until you have a quotient of zero. Here's the entire set of calculations needed to convert the decimal number 34761 into a hexadecimal number:

64540/16	=	4033 with a remainder of 12
4033/16	=	252 with a remainder of 1
252/16	=	15 with a remainder of 12
15/16	=	0 with a remainder of 15

When you've finished this series of calculations, you must convert any remainder greater than 9 into its hexadecimal equivalent. In the above problem, three remainders are greater than 9: the value 12 in the first line, the value 12 in the third line, and the value 15 in the fourth line. The decimal number 12 equates to the letter C in hexadecimal notation, and the decimal number equates to the letter F. So the remainders, converted into hex, are:

C
1
C
F

Read the above four numbers, starting from the bottom and reading up, and you have the hexadecimal number

FC1C

—which is the number we're looking for—the hexadecimal equivalent of the decimal number 64540!

DOING IT THE EASY WAY

We've compared three different number bases: the decimal system, the hexadecimal system, and the binary system. And now you know how to convert numbers from any of these three bases to any other, in either direction. Some of the conversion techniques we've covered are quite simple, others are fairly complicated, and unless you have a photographic memory, you may not remember any of them by tomorrow morning. But fortunately, you won't have to. Now that you know the principles involved, I'll tell you about an easier way to make number-base conversions. Just type and save the program presented in Listing 1 (on page 93) and you can let your computer do it for you!

By the Numbers is a menu-driven BASIC program that can convert numbers from any of the three bases discussed in this article to any other.

In next month's column, *Under the Hood,* we'll take a look inside your Commodore's main microprocessor and see what makes it tick. Then, finally, we'll be ready to start writing some programs in assembly language. ☐

SEE PROGRAM LISTING ON PAGE 93

SMALL THINGS CONSIDERED

Our congratulations to the following Commodore users, winners of subscriptions to *Ahoy!* courtesy of New York's *Small Things Considered* radio show (heard weeknights 5-8 and Saturdays 6-8 on WNYC AM83):

Pamela Rosenblatt, Zev Frankel, Meredith Jones Frummer, Melissa Rosenblatt, Matthew Higgins

THE HARDWARE INTERRUPT WEDGE

Sixty times each second the VIC and C-64 are interrupted by an internal clock. At that point they stop what they are doing and service the interrupt. They check the keyboard, update the screen and TI$, and then resume whatever they were doing when interrupted.

It is possible to "wedge" into this interrupt handler and have some of our own activities handled in background, independent of our main program. You may want to scroll a game background scene. You may want to perform some utility function such as handle the joystick or possibly take each key pressed and print it on your printer as well as on the screen. It's possible to do all this and more. The process is easy to understand and implement.

The sample programs on page 107 perform a simple operation but serve to demonstrate how to set up, enable, utilize, and, when finished, disable such a wedge. The programs are listed as BASIC loaders; a machine language assembly listing is included here so you can follow along with the explanations. The assembly listing is from the Commodore assembler.

UNDERSTANDING THE INTERRUPT

To help us understand what's going on, let's first discuss the hardware interrupt in general. We'll use the VIC and C-64 for illustration, but this applies in a general way to all microcomputers which use the 6502 or 6510 microprocessor chip. The hardware interrupt occurs 60 times each second and is processed by the 6502 (or 6510) CPU chip. All 6502's (or 6510's), on receiving a hardware interrupt, look to the pointer specified in ROM locations $FFFE, $FFFF (low byte, high byte). This pointer 'points' to the first address of the routine, which handles the interrupt. In the VIC it points to $FF72 in ROM ($FF48 in the 64). When the interrupt is recognized, the processor status register and program counter are stored on the stack and control is transferred to the routine beginning at address $FF72 (or $FF48). This routine saves the processor registers (A,X,Y) on the stack. This routine then does an indirect jump to whatever address is contained in RAM locations $0314,

Assembly Listing

```
LOC   CODE       LINE

0000             ;************************************
0000             ;* SAMPLE HARDWARE INTERRUPT WEDGE  *
0000             ;*       BY  CAL OVERHULSER         *
0000             ;************************************
0000             ;*           VIC VERSION            *
0000             ;*       (COMMENTED FOR C64)        *
0000             ;************************************
0000             ;*       'SYS828' TO ACTIVATE       *
0000             ;*      'SYS854' TO DEACTIVATE      *
0000             ;************************************
0000             *      = $033C          ;START AT ADDRESS 828
033C             VECTOR = $0314          ;SAME FOR C64
033C             ;************************************
033C             ;*          SETUP ROUTINE           *
033C             ;************************************
033C  78                SEI              ;LOCKOUT INTERRUPTS DURING SETUP
033D  A9 49             LDA #$49         ;LOW BYTE OF OUR NEW HANDLER
033F  8D 14 03          STA VECTOR       ;PUT IT IN THE VECTOR LO BYTE
0342  A9 03             LDA #$03         ;HIGH BYTE OF OUR NEW HANDLER
0344  8D 15 03          STA VECTOR+1     ;PUT IT IN THE VECTOR HI BYTE
0347  58                CLI              ;ALLOW INTERRUPTS AGAIN
0348  60                RTS              ;GO BACK TO WHERE WE WERE
0349             ;************************************
0349             ;* NEW INTERRUPT HANDLER STARTS HERE*
0349             ;************************************
0349  A5 C5             LDA 197          ;GET THE CURRENT KEY PRESSED
034B  8D FF 1E          STA $1EFF        ;PUT IT IN THE MIDDLE OF SCREEN
034E                                     ;* USE $04FF FOR C64
034E  A9 04             LDA #4           ;CODE FOR COLOR = PURPLE
0350  8D FF 96          STA $96FF        ;PUT IT IN COLOR RAM
0353                                     ;* USE $D8FF FOR C64
0353  4C BF EA          JMP $EABF        ;NOW GO DO NORMAL HOUSEKEEPING
0356                                     ;* USE $EA31 FOR C64
0356             ;************************************
0356             ;*   NEXT, PUT EVERYTHING BACK      *
0356             ;************************************
0356  78                SEI              ;LOCKOUT INTERRUPTS DURING RESET
0357  A9 BF             LDA #$BF         ;NORML INTERRUPT ROUTINE LO BYTE
0359                                     ;* USE #$31 FOR C64
0359  8D 14 03          STA VECTOR       ;PUT IT BACK IN LO BYTE
035C  A9 EA             LDA #$EA         ;NORML INTERRUPT ROUTINE HI BYTE
035E                                     ;* SAME FOR C64
035E  8D 15 03          STA VECTOR+1     ;PUT IT BACK IN HI BYTE
0361  58                CLI              ;ALLOW INTERRUPTS AGAIN
0362  60                RTS              ;ALL DONE, GO BACK WHERE WE WERE
0363             ;************************************
0363             ;*         END OF LISTING           *
0363             ;************************************
0363                    .END

ERRORS = 00000
```

FOR THE VIC AND C-64

A MACHINE LANGUAGE TUTORIAL FOR BEGINNING AND INTERMEDIATE ML PROGRAMMERS

By Calvin Overhulser

$0315. This is referred to as the vector location. When power was turned on, the start-up routine set these locations to point to ROM location $EABF ($EA31 in the 64). At $EABF (or $EA31) begins the routine to take care of the keyboard, screen, and TI$.

When this routine finishes, it restores all processor registers and returns the CPU to where it left off when the interrupt was recognized. The key to all this is RAM locations $0314, $0315. Since they are RAM, we can put our own address in them. Later, when the interrupt is recognized, we can perform our own activities and then jump to $EABF (or $EA31) to allow the normal housekeeping.

HOW TO MAKE IT WORK

Make sure any memory expanders are unplugged on the VIC. Enter the appropriate BASIC loader for your computer at the keyboard and RUN it. The BASIC loader POKEs the machine language from the DATA statements into the cassette buffer. Next, start it all off by typing SYS828 followed by the RETURN key at the keyboard. This takes us to our setup routine at $033C. Now follow along while looking at the assembly listing. We wouldn't want an interrupt to occur while we are changing its vector, so first we lock out interrupts in line 16. Then we change the vector to point to our own routine, enable interrupts, and return to the main program with RTS in line 22.

Now the fun begins! From now on, our routine, beginning in line 26, will run first when the interrupt occurs. Hit a key and watch what happens; a charcter is printed in the center of the screen. You may notice the character is not the one you press. This is because we are taking the code from the register holding the decoded keyboard matrix and storing it on the screen. These codes are different than the screen codes and this causes the strange character to appear. After we put the character on the screen in line 27, we put the code for purple in the corresponding color memory location in line 30. Last, in line 32, we jump to the normal housekeeping routine. This all occurs 60 times each second.

To disable our routine and put everything back to normal, just type SYS854 followed by RETURN. SYS854 takes us to line 37 in the assembly listing which puts the vector back to normal. Hitting the RUN/STOP and RESTORE keys will have the same effect. Make sure you disable our interrupt routine before you try to SAVE or LOAD tape, otherwise there will be a conflict since the routine resides in the cassette buffer.

APPLICATIONS

Now that we know how to use the hardware interrupt we can write routines to handle any job which we need done at a regular interval. Would you like to do sound effects or screen background motion? Would you like to do a screen dump to your printer whenever you hit a certain key? Use your imagination and all kinds of possibilities may be revealed. ☐

SEE PROGRAM LISTING ON PAGE 107

ASSEMBLER ESCAPADES

Putting Assembly Language Routines into BASIC

By Dale Rupert

Are you curious about those BASIC programs that contain line after line of numbers in DATA statements? Perhaps you know that somehow all of those numbers represent a machine language program. If you've wondered where the numbers came from in the first place, just keep reading. Enlightenment is close at hand.

This month we will continue last month's work. We will use an assembler to create a machine language program. Next, using a disassembler, we will look at that program in memory. Finally we will convert the machine language instructions into the DATA statements of a BASIC program.

If this is your first experience with assembly language, this article will be at least a two-pass process. I strongly recommend that you read it through first for an overview of all the concepts, then reread it at least once to study how the pieces fit together.

Last month we discussed the Kernal routine called SAVE. We used SAVE from within our BASIC program to put a specified portion of memory onto a disk program file. Specifically, we wrote a program to store the machine language portion of DOS 5.1 ("the Wedge") onto any diskette. We also discussed the modifications required to save a screen image onto a disk.

This month we will work with a program to save both the screen memory and the color memory data of the Commodore 64 onto separate disk files. We will start with last month's BASIC subroutine to access the Kernal SAVE routine. Then we will edit, assemble, and disassemble a machine language program to replace that BASIC subroutine. Have a look at the main program in Listing 1 (see page 91).

Lines 150 through 330 access the Kernal SAVE routine, as we discussed last month. They form a subroutine which is called from lines 60 and 90. When this subroutine is called, SADDR and EADDR contain the starting address and ending address respectively of the portion of memory to be written to a disk program file. The file's name is specified by the variable FLNAM$. The OPEN statement in line 230 replaces the SETLFS and SETNAM Kernal routines which are prerequisites for the SAVE routine. Review last month's article or refer to the description of the SAVE routine on page 293 of the *Commodore 64 Programmer's Reference Guide (PRG)*. This is the subroutine that we will replace with a machine language program.

We will come back to lines 1 and 2 of Listing 1 later. Lines 10 and 20 define the functions which calculate the most significant byte (MSB) and the least significant byte (LSB) of any integer X. Line 30 calls the subroutine at line 500 which puts a fancy design on the screen.

Lines 40 through 60 define the variables required for the SAVE routine. The filename of the screen memory is @:SCRN1. The "@:" will cause any previous version of SCRN1 to be erased before the current screen data is saved as an updated SCRN1. This is handy for debugging so that you don't have to repeatedly define new filenames every time the program is run. On the other hand, when you use this program to save your own screen images, you will want to use a new filename for each screen image.

Before putting an image onto the screen, you could use INPUT statements to specify the filename for the screen memory and the color memory. You might add lines 26 and 27 as follows to do this:

```
26 INPUT "CREATE WHAT SCREEN FILENAME?";
 FS$
27 INPUT "CREATE WHAT COLOR FILENAME?";
FC$
```

You would then replace lines 40 and 70 with these lines:

```
40 FLNAM$ = FS$
70 FLNAM$ = FC$
```

The additional statements to restore the screen image are in lines 100-140 and line 600. Line 110 clears the screen image which has just been saved to disk. Line 130 sets a flag QQ to 1. We'll see why in a moment. The color memory filename is stored in FLNAM$, and the program jumps to line 600. Line 600 loads the data from that file back into the memory location whence it came (because of the ",1" at the end of the LOAD statement.)

After the LOAD statement is done, BASIC behaves as if a GOTO 0 were next, and execution resumes at the start of the program. In line 1, the value of QQ is tested and seen to be 1, so the program branches to line 140. When this program is run for the first time, the value of QQ is 0. Consequently lines 1 and 2 are by-

passed initially.

At line 140, the QQ flag is set to 2, and FLNAM$ is given the value of the screen memory filename. Line 600 loads the screen file data back into the proper area of memory which is now at locations 1024 through 2023, and the image reappears on the screen. As before, execution continues from the beginning of the program. Since QQ now has a value of 2, line 2 causes the program to end. If you wish to do something else with this screen image, simply modify line 2 to jump to your own routine.

THE ASSEMBLER

Our next goal is to write an assembly language routine to duplicate the functions of the BASIC statements in line 240 through 310 of Listing 1. First you must have an assembler program. I used the *IEA Instant Editor Assembler* from Robin's Software (Bloomington, MN). It is a reasonably priced editor, assembler, and monitor package that is self-contained and relatively easy to use. If you are using another assembler, the procedures in the following discussion will be very similar.

The *IEA* assembler does not require a separate editor program to create the assembler source file. Instead it adds some commands to the BASIC screen editor. Thus you create an assembly language routine much the same way you do a BASIC program. Simply type each assembler statement on a separate line. Although *IEA* allows the BASIC and assembler routines to be contained within the same file, we will create the assembler routine separately.

Look at the statements in Listing 2 (page 92).

With the exception of the periods at the start of each line, lines 210 through 340 were typed just as shown. (Lines 10 through 30 and the initial periods on each line were added later in order to create the magazine listing.)

The assembler takes this "source code" which consists of somewhat Englishlike statements and translates it into "object code" which is the actual machine language numbers that the processor understands. Some of the statements in this source code translate directly into machine code, while others are special instructions for the assembler, called "pseudo-ops."

First we will discuss the 6510 microprocessor instructions in the source file. Then we will look at the pseudo-ops. Pages 232 and 233 of the *PRG* list the 6510 instruction set. Using that list, you can see that lines 230 and 260 through 320 of the source listing contain actual 6510 instructions.

Line 230 tells the processor to JuMP to the line labeled ":ENTRY". Each label is preceded by a colon. The line numbers in Listing 2 have been chosen to correspond as much as possible to lines performing similar functions in Listing 1.

The accumulator is frequently used as an intermediary since the 6510 microprocessor cannot directly move data from one memory location to another. Line 260 loads the accumulator (LDA) with the value stored in the memory location labeled ":SADDR". Line 265 takes that value in the accumulator and stores it (STA) directly into memory location $FC, which, you may recall from last month, is a page-zero location available to us. The "$" indicates a hexadecimal value. Similarly the value in memory location ":SADDR + 1" is loaded into the accumulator and then stored into page-zero location $FB in lines 270 and 275. (We will see what those memory values are a little later.)

Line 280 is a "load immediate" instruction as indicated by the "#" in front of the $FB. Rather than treat the $FB as an address as it did with ":SADDR" above, the microprocessor will put the value of $FB directly into the accumulator.

Lines 290 and 300 take whatever values are in memory locations ":EADDR + 1" and ":EADDR" and load them (LDX, LDY) into the X and Y registers. Glance back at Listing 1, and see that these assembly language instructions duplicate the functions of the BASIC instructions in lines 260 through 300.

The JSR instruction in line 310 is comparable to a GOSUB statement in BASIC. It tells the processor to call the subroutine whose address is stored in the memory location labeled ":SAVE". You can see that the assembler memory location labels are analogous to variable names in BASIC.

At the end of the Kernal SAVE routine is an RTS (Return from Subroutine) instruction which transfers control back to our program at line 320. The RTS instruction in line 320 returns control back to BASIC. Notice taht Listing 1 doesn't need such a statement since the SAVE routine returns directly to BASIC. Listing 2 creates a machine language program which calls another machine language Kernal routine, whereas Listing 1 is a BASIC program that calls the Kernal routine.

PSEUDO-OPS

The instructions in Listing 2 that we have not yet discussed are the pseudo-ops. They look somewhat like assembly language mnemonics (LDA, RTS, etc.) but they are not among the instructions recognized by the 6510 microprocessor. The pesudo-ops are instructions to the assembler.

The ORG (Origin) instruction tells the assembler where the program is to reside in memory. We've chosen $C000 for the start of our routine. The next pseudo-op is the .EQ (Equate) in line 240. This is equivalent to the LET statement in BASIC. It assigns the value $FFDB to the label :SAVE, just as we did in line 240 of Listing 1. The .EN instruction in line 340 tells the assembler where the end of our assembler source code is. These three pseudo-ops do not actually cause any data to be put into memory.

The pseudo-op .DB (Define Byte), on the other hand, does cause data to be stored in memory. The location in memory for the first of the two bytes $04 and $00 will be referred to by the label :SADDR, and :EADDR

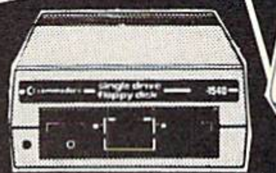
USER FRIENDLY PRICES!
(BELOW WHOLESALE)

COMPUTER CENTERS OF AMERICA

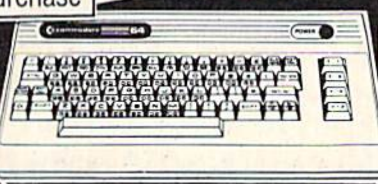
FREE Dustcover for Commodore 64 with any Hardware Purchase



commodore
1101 LETTER QUALITY PRINTER
274.95

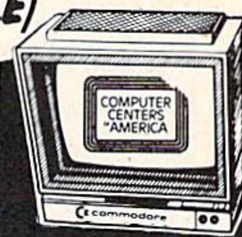


commodore
VIC 1541 DISC DRIVE
169.95



commodore 64
149.95

NEW COMMODORE 128 279.95



commodore
1702 MONITOR
174.95

CARDCO PRODUCTS IN STOCK & ON SALE!



Monitors

BMC Color 149.95
Pan 1300 RGB & Comp. 289.95
BMC Green 69.95



Modems

Comm 1660 .... 84.95
Mighty Mo .... Call
Telesonic .... Call
Hes II .... Call



Cassette Recorders

Comm CN-2 .... 59.95
Third Party 39.95



Printers

Gemini SG 10/15 New! 209.95
Gemini SD 10/15 New! Call
Gemini SR 10/15 New! Call
Gemini SB 10 New! .. Call
Gemini 15X 399.95
Silver Reed 400 .... 249.95
Silver Reed 500 .... 379.95

Interfaces

Cardco G + ... Call
Microworld .... Call

Diskettes



Nashua ... 10.95
Scotch 14.95
Maxell 14.95
Polaroid 13.95



Indus GT Disc Drive for Commodore
249.95



New! MSD Super Disc II
Now On Sale
449.95
Single Drive 249.95

WE'RE PROUD OF OUR PRICES—SATISFACTION GUARANTEED

TO ORDER: CALL TOLL FREE **1-800-631-1003**

IN NEW YORK STATE CALL TOLL FREE 1-800-221-2760 OR 516-249-2120

Check, Money Order, MC or VISA accepted/No additional charge for MasterCard and Visa/Call for shipping and handling information/NYS residents add applicable sales tax/Prices and availability are subject to change without notice/All factory fresh merchandise with manufacturer's warranty. Dealers Welcome. Call for recent price reductions and new rebate information.

VISA · MasterCard

COMPUTER CENTERS AMERICA
or Write: Computer Centers of America
81 Terminal Dr.
Plainview, NY 11803

We're Reliable! 18 years of service.

TOUCH TABLETS

Koala Touch Tablet-D 64.95
Koala Touch Tablet-Cart 69.95
Muppet Learning Keys Call

ACCESS

Neutral Zone-D/T 20.95
Spritemaster-D/T 20.95
Beachhead-D/T 20.95
Master Composer-D 23.95

SPINNAKER

Adventure Creator-Cart 21.95
Aerobics-D 25.95
Aegean Voyage-Cart 21.95
Alf in the Color Caves-C 21.95
Alphabet Zoo-Cart 20.95
Bubble Burst-Cart 21.95
Cosmic Life-Cart 20.95
Delta Drawing-Cart 21.95
Facemaker-Cart 20.95
Fraction Fever-Cart 20.95
Grandma's House-D 20.95
Jukebox-Cart 20.95
Kids on Keys-Cart 20.95
Kidwriter-D 20.95
Kindercomp-Cart 17.95
Ranch-Cart 20.95
Rhymes/Riddles-D 17.95
Search/Amazing Thing-D 24.95
Snooper #1-D 24.95
Snooper #2-D 24.95
Story Machine-Cart 20.95
Trains-D 24.95
Up For Grabs-Cart 20.95

COMMODORE

Program Ref. Guide 19.95
Assembler-D 17.95
Easy Finance I, II, III, IV-D 19.95
Easy Calc-D 64.95
Easy Mail-D 17.95
Easy Script-D 39.95
Easy Spell-D 19.95
Logo-D 49.95
The Manager-D 37.95
General Ledger-D 37.95
Accts. Rec.-D 37.95
Accts. Pay.-D 37.95
Magic Desk-D 52.95
Zork I, II or III-D 29.95
Suspended-D 29.95
Starcross-D 29.95
Deadline-D 29.95

CBS SOFTWARE

Argos Expedition-D 29.95
Charles Goren's Bridge-D 49.95
Coco Notes-D 19.95
Ducks Ahoy-D 23.95
Ernie's Magic Shapes-D 23.95
Mastering the SAT-D 104.95
Movie Musical Madness-D 24.95
Murder by the Dozen-D 23.95
Peanut Butter Panic-D 24.95
Sea Horse Hide 'n Seek 24.95
Success Decimals (Add/Subt)-D/T 14.95
Success Decimals (Mult/Div)-D/T 14.95
Success Fractions (Add/Subt)-D/T 14.95
Success Fractions (Mult/Div)-D/T 14.95
Timebound-D 24.95
Webster Word Game-D 24.95

WICO Joysticks



The Boss 12.88
The Bat ... 14.88
3-Way 19.88

commodore NEW! Products

801 MCS Color Printer **269.95**

C-16 Computer 79.95
C + 4 Computer 189.95
801 Printer 149.95
802 Printer 179.95
803 Printer 174.95

identifies the memory location of the first of the two bytes $07 and $E8. The second bytes are at (:SADDR + 1) and (:EADDR + 1) respectively.

If we were to assemble this program by hand, we would have to figure out how many bytes long the JMP instruction in line 230 is in order to know where in memory the $04 and the $00 will reside. The assembler takes care of that for us. It allows us to refer to the memory locations by means of the labels we have chosen. Consequently we don't even need to know the actual numerical values.

Since assembly language is a lower-level language than BASIC, comments are even more crucial for debugging and for figuring out tomorrow what the routine you wrote today was supposed to do. A semicolon defines the start of a comment in the source code. Anything after the semicolon on a line is ignored by the assembler.

Right now you may be wondering what we do with the source code we've created. First we save the source code just as we would save any BASIC program we have typed in. Then we assemble it.

With the *IEA* assembler, the instruction .A puts the assembler into action. The mnemonics are translated into their numeric values. The values following the .DB pseudo-ops are inserted in their proper places. Also a symbol table is created which shows the numeric values assigned to each of the labels. Typing .X with the *IEA* assembler displays a symbol table as shown in lines 40 through 70 of Listing 3 (page 92).

Notice that :SADDR begins at $C003. The ORG statement specified that our program was to begin at $C000. From this we can conclude that the JMP instruction requires three bytes, residing from addresses $C000 to $C002. The label :END has a value $C01D, so our program will occupy memory locations $C000 through $C01D. Actually the .EN pseudo-op creates a $00 byte at location $C01D which corresponds to a BRK (Break) instruction. Refer to page 256 of the *PRG* to see the hex values (called the op-codes) of all instructions.

Now that we have edited (created) and assembled this program, we can save it to disk as an executable file if we wish. Using the .S command saves a specified range of memory under a filename that we choose. The program saved is called the object code. In order to see what the object code looks like, with the *IEA* assembler we must use a separate monitor program.

THE MONITOR

A monitor program allows you to readily peek at data and instructions stored in memory. A monitor will typically contain a disassembler which translates the numbers it finds in memory back into the mnemonics that we used with the assembler. Generally a disassembler cannot create labels since those are not stored in memory as part of the object code. (A symbolic debugger will allow you to assign labels and refer to them during debugging, but we won't use one here.)

Listing 4 (see page 92) shows the result of disassembling our object code as it resides in memory.

I have added the numbers in the left column for the magazine listing. They correspond to the numbers in Listings 1 and 2. The disassembled object code is in essence what the microprocessor sees when it executes our program (yes, we know the values in memory are really 0's and 1's rather than hexadecimal, but binary is too cumbersome for our brains).

The second column of numbers indicates the memory addresses where each instruction resides. The next one, two or three hex values are the op-codes for each instruction. The last columns contain the mnemonic instructions corresponding to the op-codes.

For example, beginning at memory location $C000 are the three bytes $4C $07 $C0. When the processor sees the $4C instruction, it recognizes it as a JMP statement. The processor expects the next two bytes to represent the address to which it will jump. Those two bytes are stored in the LSB, MSB format we have already discussed. Consequently the processor jumps to address $C007 in order to fetch its next instruction.

Some instructions are only one byte long (RTS) and others may be two or three bytes long. Part of the processor's job is to determine from the op-code how many bytes make up each instruction. If you glance at the description of the LDA instruction on page 245 of the *PRG*, you will see that it may be either a two or three byte instruction. How does the processor know? Very simply, each form of the instruction has a separate op-code.

Op-code $A9 implies a two-byte instruction, and op-code $AD is part of a three byte long instruction. The assembler mnemonic is LDA in both cases, but the processor sees them as two different instructions. In the first case (line 280 in Listing 4), the processor treats the next byte ($FB) as data which is immediately loaded into the accumulator. In the second case (line 270), the next two bytes ($04 $C0) are treated as an address ($C004) from which the processor retrieves a byte of data to be stored in the accumulator.

The disassembler treated our data in locations $C003 through $C006 as if they were op-codes. The $04 and $07 are not valid op-codes whereas $00 and $E8 just happened to be valid. These values correspond to the screen memory starting and ending addresses that will be used by the Kernal SAVE routine. Our BASIC program will POKE the correct data into these memory locations before this program is executed. The processor will not mistakenly execute this data since the JMP instruction at $C000 branches around it.

To test your understanding of this disassembly listing, see if you can figure out what value will be loaded into the Y register when the instruction at memory location $C016 is executed. If you said $07, you are correct. LDY $C005 tells the processor to take the value from location $C005 and put it into the Y register. According to line 253, the value in location $C005 is $07.

An integrated business software package for the C-64™ for $1.59 a Program? Yes.

The Intelligent Software Package for **C-64™**, **VIC™**, **SX-64™**, **Plus 4™**, **16™**, **'B'-Series™**, and **PET/CBM™**. No games, no fancy packaging, and you can't buy it anywhere but here. But: 22 powerful, useful programs (totalling over 51 *pages* of source code) that will put your Commodore to work right now, in your home or office; all on disk or tape at the ridiculous price of **$35.** Includes:

Database: A complete multi-keyed fixed-record-length data base manager. Sort or select (using all relational operators:, = >, <2, AND, OR, NOT, wild card) on any field, perform computations on numeric fields. Any operation can be performed on all, or only selected records. All fields completely user-definable. Can be used for any number of tasks, including accounting, mailing lists, inventory control, record, tape, or book cataloging, expense account manintenance, or as an electronic rolodex. Even if you use your Commodore for nothing else, this program alone might justify its expense.
Word Processor: The orginal word processor for the VIC-20, W/P is now available for all CBM models. A full-featured menu-driven word processor including very fast file commands (including a disk catalog), screen editing, text locating and full control over margins, spacing, paging, indentation, and justification. "*. . . well done and highly functional . . . Provides an excellent alternative to the high priced word processors . . . this is an excellent buy. Highly recommended.*" — Midnite Software Gazette. "*Provides good basic features.*" — Compute's Gazette.
Copycalc: An electronic spreadsheet. Turns your Commodore into a visible balance sheet; includes screen editor. "*Excellent program for budgeting, estimating, or any math-oriented use . . . well worth the money. Highly recommended.*" — Midnite Software Gazette.

Also Included: **ReportGen, ReportMerge** (interface W/P with Database to create form letters, statements, invoices, mailing labels, other reports.); **Baseball Statistician** (compiles batting statistics for a baseball league); several W/P utilities, including **Index** (indexes W/P's text files); several Database utilities, including **DBmerge** (facilitates multi-file database applications.), AND **DBStat** (analyzes D/B files); a programming utility, **ASCII,** which converts text files (program listings) into program files; also **Checkbook**; **Inventory**; **Paper Route**; **Loan Analysis**; **Breakeven Analysis**; **Depreciation**; **Labeler**; more.

Versions of the package are available for any and every Commodore computer having a minimum of 10k RAM (does it say 'Commodore' on it? How about on the box it came in?) Add $3 if you have an 8050 disk drive. All programs will fully support tape, disk, and printer. The package is available on disk or tape (disk version only for Plus4™ and 16™.) Price includes documentation and shipping within USA and Canada; all other countries, add $5 (no personal checks from outside USA). No charge cards. For C.O.D., enclose $5 fee; Calif. residents add 6%. This ad is the catalog.

Intelligent Software

Quality Software since 1982

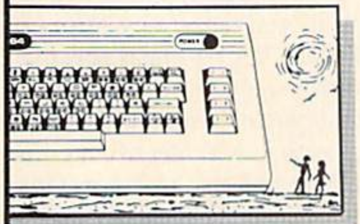
Box A Dept. A
San Anselmo, CA 94960

Reader Service No. 230

C-64 UTILITIES

COPY CLONE with Utilities Copy protected disks in 5 minutes or less. Works with **MOST** software. Will not damage disk drive. **Over 20 utilities,** including professional disk drive analysis • 30 second formatting • sophisticated disk editor • and much, much more. **$19⁹⁵**

Make your C-64 Grow!



"Who Needs IBM?"

POWER PLUS

Utility adds 40+ commands, doesn't use BASIC memory • 100% machine. • SCREEN DUMP to printer • UN-NEW • MACHINE LANG. MONITOR • Adds BASIC commands AUTO LINE NO., RENUMBER, FIND, CHANGE, DELETE, more. **$19⁹⁵**

14 ACTION GAMES When you order both programs **FREE**

MSD DUAL DRIVE ONLY $525 plus $10 p/h

Send check or money order (include $2.00 postage & handling) CA residents add 6% sales tax to:
VISA MASTERCARD

EDUCOMP
2139 Newcastle Ave., Ste. A
Cardiff, CA 92007
(619) 942-3838

HOT ACCOUNTS™

The most highly acclaimed bookkeeping system for the Commodore 64®

A Superb! An exceptional program that outshines all others.
RUN Magazine, Nov. 1984

HOT ACCOUNTS™ teaches you bookkeeping as it automatically performs dozens of full-service accounting functions. It is a complete system that will instantly organize your records and receipts and supply charts, formal statements, and budget information. Now your records can be accurate, up to date and confidential. ▪ Tracks charge cards, checkbooks, all transactions ▪ Requires no prior knowledge of accounting ▪ Home or small business ▪ Improves your financial image ▪ Cost less than 1 hour with a C.P.A.

HOT ACCOUNTS™ Bookkeeping System on Diskette

For **RUSH** delivery, send check or money order for $64.95 to:
Computer Ed. Software
1002 Brookes Ave. W., Ste 100
San Diego, CA 92103

Credit card orders, call **toll-free**
1-800-621-0852 ext. 238
VISA MASTERCARD

P.S. Use HOT ACCOUNTS™ to do your taxes and the price is tax deductible!

BONUS OFFER! Order before **July** 15th and receive a full featured, 60 function **word processing** & 3-D **business graphics** package as our **FREE** gift to you.

Reader Service No. 229

Now what value will be stored in page-zero location $FB by the instruction at location $C00F? If you said $00, you're right! Notice that the accumulator contained the value which it got from $C004 in line 270. The value in $C004 is $00.

BACK TO BASIC

We can now modify our original program to implement the machine language routine we've just discussed. From the disassembler output, we easily create the DATA statements in Listing 5 (see page 92). The statements in Listing 5 will replace the corresponding lines in Listing 1.

The data is POKEd into memory by the statements in lines 190 through 220. Once that is done, memory locations $C000 through $C01C look the same as they did after we assembled our source code. The first time this subroutine is called (by line 60), the machine language data is read and stored in memory. The second time it is called (by line 90), lines 190 through 220 are bypassed.

Lines 240 through 290 put the appropriate screen or color memory starting and ending addresses into the proper RAM locations (which we determined from the symbol table in Listing 3 and from Listing 4).

Finally the SYS statement in line 310 sends the processor to our machine language routine at address 49152 ($C000) where the specified memory data is written to a disk file. As in Listing 1, the filename is established by the OPEN statement in line 230.

There is really no advantage to using the machine language version of this subroutine compared to the BASIC version. The machine language version has been written to exactly duplicate the BASIC one. The Kernal SAVE routine takes the same amount of time to execute regardless of which routine calls it, so we have no speed advantage in this particular case. This has been primarily an academic exercise.

Hopefully you now have a better understanding of how machine language routines may be incorporated into BASIC programs. Perhaps you have learned something about editing, assembling, and disassembling. This article has provided a model which may be useful for creating and calling your own machine language routines from BASIC. If nothing more, at least you have two versions of a program for saving and restoring screen images on disk.

The *IEA Instant Editor/Assembler* is $39.95 + $2 p&h from Robin's Software, 10349 Zinran Circle, Bloomington, MN 55438. Mark Robin mans the assembly language hotline at 612-944-8654 from 6 to 10 p.m. (CST) seven days a week to assist Commodore 64 assembly language programmers. ☐

SEE PROGRAM LISTINGS ON PAGE 91

RAID

For the C-64

By Larry Coats, with Noel Nyman

You are making a raid on Heartland to capture as many hearts as you can. But beware of the butterflies. One touch and you'll lose one of your lives. Touch too many and you lose the game.

This game uses a joystick to move a playing piece through a simple maze, capturing hearts by touching them. Floating around in the maze are butterflies. You must avoid them and collect all the hearts to exit from the maze. If you make it, you score extra points and get a new maze.

Up to nine people can play, taking turns with the joystick. There are nine levels of play as well. The game may seem simple at first. But looks can be deceiving! Machine language is used for the maze action to provide fast movements and quick joystick response.

Two programs are used to LOAD and RUN *Raid*. Type in the *Disk Boot* on page 94 if you use a disk drive. Use *Cassette Boot*, also on page 94, if you have a datasette. Then type in the main listing. SAVE the program with the name *Raid* so the boot program can LOAD it properly.

Since there are many DATA statements, we've included a "checksum" to help you detect any errors in typing. The last number in each DATA line is negative. Be sure to type in the minus sign.

When you first RUN *Raid*, you may see an error message on the screen. The message will identify the line number of a DATA statement that has an error. Once you've found all the errors, the program will display the maze and ask you for the number of players and level of difficulty.

You can remove the REM statements in lines nine through twenty-five if you wish. Line #2140 displays the updating address on the screen during the machine language load. The program will LOAD faster if you leave out the two PRINT statements on this line. Don't take out the POKE statement!

We hope you enjoy this simple but challenging game. ☐

SEE PROGRAM LISTING ON PAGE 94

DOS PLUS

Continued from page 39

than the start of BASIC. For instance, you might have a hi-res screen you wish to save, or a block of sprite data that you want to write directly to disk. Give your data block a filename, enter its starting address and ending address+1, and *Dos Plus* will do the rest. No more loading a monitor to save machine language programs! You may reload such saved files by typing LOAD "filename",8,1 <return>. You could even save BASIC ROM itself, if that were your passion. Notice that the file saved is automatically verified for you.

This one is used several ways:

```
*FILENAME <RETURN>
```

or

```
*"FILENAME" <RETURN>
```

or by entering the asterisk directly in front of a filename in a directory listing. This is the SINGLE-FILE COPY command structure. After entering it, you will be prompted for a destination. Single drive users should respond by entering '8'. The file will now be copied into memory from the disk that is presently in the drive without overwriting any program you might have in memory. The size of file that may be copied is limited only by the amount of memory available between the next even page boundary above the end of any program in memory and the top of BASIC. If, during the copy process, the file exceeds the top of BASIC high memory, the copy will abort with 'FILE TOO BIG.' If you are going to copy a very large file it would be wise not to have a program in memory at the time of copy. After the entire file has been brought in from disk you will be asked to insert the destination disk and press RETURN. The entire file will then be written to the destination disk. The file will be auto-verified at that point (unless you have defeated that function). If you wish to abort the copy process, you can do so by pressing RUN/STOP at the 'INSERT DESTINATION DISK' query. Do not attempt to copy relative files with this function. Sequential, user, and program files may all be copied but will be written as program files. If you use this function to copy a non-program type file you may use the @:D (block dump/modify) command to change the file-type identifier in the directory. If you are using two disk drives you may enter '9' to the 'TO ? 8/9' prompt. Be careful that drive '9' is present or the computer will 'hang.' After the copy process is complete you will be logged to drive 9.

Since drive logging has been mentioned, note that if you are using two disk drives, one as drive 8, the other as drive 9, you may use *Dos Plus* to work with both. To change the drive you are currently logged to, enter:

```
@8 <RETURN>
```

or

```
@9 <RETURN>
```

as appropriate. After logging to a given drive, all future disk operations with *Dos Plus* will be performed on that drive. To work with the alternate drive you must log to it with the correct @<number>. Also note that this is automatically done when you use the copy function (*) and enter the appropriate drive number to the 'TO ? 8/9' prompt.

Want to verify a program that is at a location other than the start of BASIC? Type:

```
@:V'FILENAME',$<START ADDRESS><RETURN>
```

This is primarily useful if you have disabled the auto-verify function of *Dos Plus*. Since verifying large saves can be time consuming, you may wish to disable the auto-verify on saves or copys from within *Dos Plus*. You do this by changing the DOS.BOOT program which loads DOS PLUS. Alternately, you may POKE the appropriate location as is listed in Table 1. The @:V (verify) command enables you to verify saves or copys as you see fit.

Some additional commands:

@ –Read disk error channel

/filename – Load a program at start of BASIC
%filename – Load a program at its load address
←filename – Save a program from start of BASIC (will auto-verify unless disabled)
↑ filename – Load & Run a program at start of BASIC
#<nnnn> – Give hex number for decimal input
$<nnnn> – Give decimal number for hex input
@Q – Disconnect *Dos Plus*

These commands work as they do in previous Wedge programs. The hex and decimal number conversions use 16-bit arithmetic, meaning that the largest values returned are 65535 and $FFFF respectively. Right justification is used for hexadecimal input. This means that $ABCDE will be taken as $BCDE, etc.

For the hard to please among you and the true hacker types, three command structures have been supplied that are null routines. These are: @:1, @:2, and @:3. All three merely print a carriage return and jump to BASIC at $E386. You may revector these commands to your own custom routines by changing the last three jump addresses at the very end of the *DOS Plus* machine language to the addresses of your routine. Characters entered with these commands will be in the system input buffer at $0200 and are terminated with a zero byte. Your routines should return to BASIC with a JMP $E386 instruction, not an RTS to DOS PLUS.

The DOS.BOOT program contains several POKEs that are user modifiable. These addresses and their functions are outlined below.

POKE 53230,001 : 1=auto-verify 0=don't auto-verify
POKE 53012,008 : default drive at power-up
POKE 51329,096 : 96=UPPER CASE Printer Sec. Addrs. 103=lower case Printer Sec. Addrs.
POKE 53040,018 : 18=reverse video on directory header 146=no reverse video
POKE 53063,018 : same as above
POKE 50999,144 : color of hex numbers in DUMP/MODIFY
POKE 51069,144 : same as above
POKE 51057,005 : color of ASCII digits in DUMP/MODIFY (this combination looks best on my amber Taxan. Alter to your tastes – use ASCII number for color code, not Commodore color codes, i.e., 5=white, 28=red, 144=black, etc.)

By POKEing these locations you can, with the appropriate values, alter *Dos Plus* to your own tastes. Any, or all, of these locations can also be changed directly from the keyboard should you so desire. Merely POKE the location with the value you wish to substitute. This will save you from having to run the loader program to make the changes.

Well, there you have it – a balanced set of tools all wrapped in one tight package that won't steal precious space from your BASIC programs. Since *Dos Plus* takes up all but two of the locations at $C000 - $CFFF, you will most likely find a conflict in memory usage with some other utility that you like to use. Experienced machine language programmers will be able to get around this by relocating these utilities and patching them to *Dos Plus* with the supplied hooks. If it is beyond your present programming skills to implement such a patch, you will find the inconvenience to be more than outweighed by the usefulness of *Dos Plus*. Who knows? After a time you may wonder how you ever got along without it! Shortly after I wrote *Dos Plus*, I completely destroyed the directory tracks on a CP/M disk on which I had the only copy of a compiler I was writing. Using the BLOCK DUMP/MODIFY function of *Dos Plus*, I was able to reconstruct the directory tracks and thus recover three months' worth of work.

Writing *Dos Plus* has been an educational experience for me. I hope that using it will prove to be a pleasurable and educational experience for you.

For those who wish to patch their own machine language routines to *Dos Plus*, or are merely curious or want to learn more about assembler, I will supply copies of the source code for *Dos Plus*. Since this amounts to over 30 pages of heavily commented hardcopy I will have to ask that you help cover the cost of paper. Send $1.50 and a self-addressed, stamped manila envelope to: Dennis Muscatelli, R.D. #1 Box 543, Morrisdale, PA 16858.

I wish to express my deepest thanks to Professors William Englebret and Gene Schlossberger at The Pennsylvania State University for their critical comments and assistance with debugging. ☐

SEE PROGRAM LISTINGS ON PAGE 98

ON-COURT TENNIS
Gamestar
Commodore 64
Disk, joystick; $29.95

Anyone who doesn't think computer games have come a long way since *Pong* ought to try a few sets of *On-Court Tennis.* It's hard to imagine two games based on the same sport being more different.

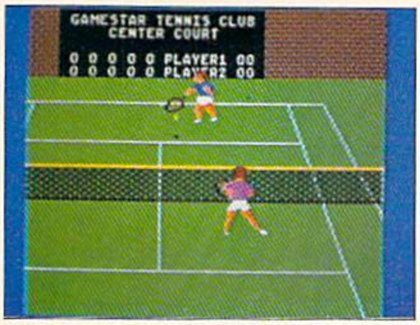
On-Court Tennis features fluid animation, highly sophisticated computerized opponents in the solitaire mode, and true-to-life strategy. It's a far cry from rectangular racquets hitting a square "ball." In fact, the hardest part about learning to play this sports simulation is the way it shatters preconceptions.

The most radical change is that the joystick is used to select the desired stroke, not move the athlete around the screen. In effect, *On-Court Tennis* is a strategy, as opposed to an action, game.

The display shows the court in three-quarter perspective, with the serving player at the bottom of the screen. As the onscreen character tosses the ball into the air, the computerist moves the joystick to start the volley. The direction in which the stick is pushed determines the placement of the shot. Hitting the action button produces an especially hard serve.

The computer automatically positions the racqueteer into position to return a shot. The main challenge, therefore, is timing the swing and choosing the specific stroke. The gamer can pick a flat shot, lob, topspin shot, slice, or drop shot. Whether the ball goes left, right, or straight ahead depends on when the player swings.

The meticulously accurate ball-shadow isn't just window dressing. It's the principal guide to timing a racquet stroke. This can be tricky, since the shadow is in a different position, relative to the ball, depending on whether the onscreen player is in the far court or the near court.

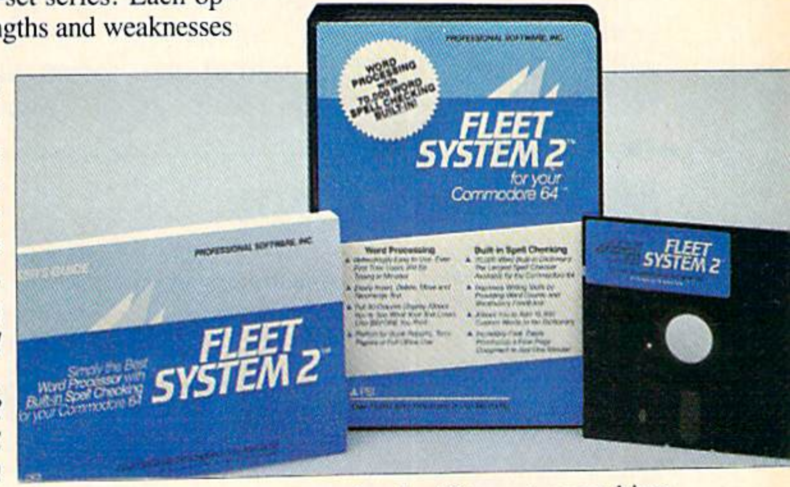


On-Court Tennis: *an outright winner.*
READER SERVICE NO. 191

Four robot foes are ready to battle the computerist in a single set or a three- or five-set series. Each opponent has strengths and weaknesses and favors a particular style of play. "Ivan Messier," for example, plays the baseline and boasts a terrific serve. "Jimmy Orr," on the other hand, roams the entire court and depends most heavily on a superb forehand. All four are programmed to vary their skill as the human player gains mastery of the game.

The playing surface also affects the difficulty level. The ball bounces true on a hard court, goes faster on grass, and slows on clay.

Novices are strongly urged to begin by trying one set on clay using the baseliners. *On-Court Tennis* is a very demanding game even at its easiest setting, and the first few rounds of play are bound to be error-filled and highly frustrating. This is one game that definitely repays the investment of a couple of practice sessions. If all else fails, watch the computer play itself to get some hints.



Fleet System *can check any document against a 70,000 word dictionary on the flip side of the disk (without exiting the program) and highlight all suspect words.*
READER SERVICE NO. 192

On-Court Tennis is truly a landmark computer entertainment program. It takes a fresh look at a subject, video tennis, which many considered totally played out. This outstanding disk proves them wrong.

Gamestar, 1302 State Street, Santa Barbara, CA 93101 (phone: 805-963-3487). *—Arnie Katz*



Trolls and Tribulations *(see page 71).*
READER SERVICE. NO. 193

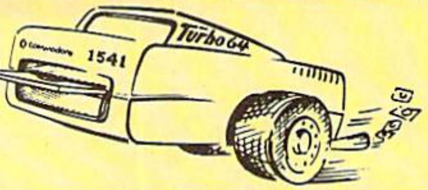
FLEET SYSTEM 2
Professional Software, Inc.
Commodore 64
Disk; $79.95

What kind of word processor can you expect from a company whose latest program was *Trivia Fever?* Before tossing off a flip one-liner, consider this: the company is Professional Software, whose venerable

Reader Service No. 178

TURBO 64

$19.95



Turbo 64 will turn your 1541 into a super fast and efficient disk drive. Loads programs five to eight times faster, works with 99% of your basic and machine language programs. The master disk allows you to put unlimited Turbo 64 boot copies on all your disks. This disk also includes:

Auto Run Boot Maker

Auto Run Boot Maker will load and auto run your basic or machine language programs.



$19.95

DMS-Errors 20,21,22,23,27 & 29
Format Single Tracks
Read Disk Errors

½ Track Reader-read and select ½ track.

½ Track Formatter-Format a disk with ½ tracks. This is where the next protection schemes are coming from.

Drive Mon-Disk Drive assembler/disassembler. For your 1541.

The Doc-Disk Doctor that reads code under errors.

Sync Maker-Place a sync mark on **any** track out to 41. Also used for protection.

Sync Reader-Check for Sync bits on **any** track out to 41.

Change Drive No.-Changes drive number (7-30).

Disk Logger-Finds starting track, sector; start and end addresses.

Disk Match-Compare any two diskettes. Byte for Byte.

New Wedge-Easier to use DOS wedge.

ID Check-Check ID's on **any** track.

Unscratch-Restore a scratched file.

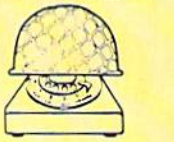
View RAM-Visual display of the free and used sectors on a diskette.

Read/Write Test-1541 performance test.

Repair a track-Repair a track with checksum errors. Reads code under errors and restores track.

Fast Format-Format a disk in just 10 seconds (with verify!).

WAR GAMES AUTODIALER



1-**Auto Dial** will automatically dial a set of numbers you choose.
2-**Review Numbers** will review numbers that were answered by a computer.
3-**Save Numbers** will save numbers where a computer answered.
4-**Hardcopy of Numbers** will print out list of numbers where a computer answered.
5-**LOAD Numbers** will load in numbers to continue where it left off.
6-**Continue** will pick up dialing where it was interrupted.

$29.95

KARTRIDGE KRACKER

NOW you can own this unique and powerful tool which will allow you to dump the contents of 8K and 16K cartridges onto disk! But what's really great is that you can also RUN the cartridges programs without plugging in the cartridge! the KRACKER gets YOU INSIDE the cartridge! Put all your favorites on disk and get rid of the clutter. This package provides your with the software and hardware needed to get started. Program on disk included. (Some cartridges require use of external RAM not included)



$44.95

SOFTWARE PROTECTION HANDBOOK

3rd Edition

THIRD EDITION! NOW AVAILABLE!

If you're **tired** of being harassed by **protected software** and too many **copy** programs, then this is the book for you! This 224 page manual covers the gambit from legalities to protection methods to step-by-step back up procedures. Now you can learn both how to **protect and unprotect** software! The techniques covered include copying cartridges to tape or disk, tape protection, and disk protection. Disk protection covers error no.'s 20, 21, 22, 23, 27 and 29 **plus** single track formatting, header modification, header swapping, half track reading and writing, reading and writing modified bit densities, formatting illegal tracks/sectors, sync writing and **more!** The Third edition **explains,** tells how to **detect** and how to **write** them with included software. Eleven useful utilities and many protection listings! Our disk analysis programs reveal the protection methods used on your originals. A diskette with all software is available for a minimum extra charge. This may not be the only book your should have for the C-64, but it is certainly **the one** book you should not be **without!**

C64 Book only .... $19.95 US
Book & Disk of all programs .... $29.95 US
Vic 20 book . . . Cart. & Tapes only .... $9.95 US

THIS MANUAL DOES NOT CONDONE PIRACY

*SHIPPING: $2.00

The Original Book of Facts

UNLOCK THE MYSTERIES OF THE COMMODORE

—Simple Instructions
—Easy step-by-step examples
—Tricks you can do your first day
—Introduction to color, sound and much more
—Includes a routine to protest your own programs
—NO MATTER what your experience, the Original Book of Facts is for everyone!

$6.99

The Advanced Book of Facts

EXPLORING YOUR COMMODORE

—How to use a machine language monitor
—Tips on how to unprotect your software
—Advanced protection for your own programs
—Full size reference charts
—How to design your own menus and more...

$6.99

BULLETIN BOARD

Set up and operate your own bulletin board with one or two disk drives. This one has all the features and you can customize it easily yourself. ·

1-RUN MEGASOFT BBS
2-CREATE MEGA FILES
3-ADD TO SYSOP'S CORNER
4-NEW SYSOP'S CORNER
5-READ MESSAGES
6-SCRATCH MESSAGE
7-CYCLE MESSAGES
8-READ SYSOP MESSAGES
9-WRITE OPENING MESSAGE
10-READ LOG
11-CYCLE LOG
12-READ DOWNLOAD FILE
13-SCRATCH DOWNLOAD FILE
14-ADD TO OTHER SYSTEMS
15-CREATE OTHER SYSTEMS

$59.95

Enclose Cashiers Check, Money Order, or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders. Canada orders must be in U.S. Dollars. VISA — MASTER CARD — COD

Programs for C-64 $2.00 S & H on all orders.

Software Submissions Invited.

MegaSoft Limited

P.O. Box 1080, Battle Ground, Washington 98604

Phone (206) 687-5116 • BBS 687-5205 After Hours Computer to Computer

Reader Service No. 178

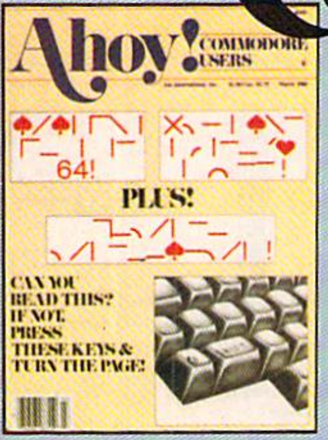
Back Issues of Ahoy!



ISSUE #1—JAN. '84 $4.00
64 v. the Peanut! Telecommunications! Tales of the Commodore! And ready to enter: Multi Draw 64 Graphics System! Interrupt Music Maker/Editor! A Peek at Memory! Sequential Files!



ISSUE #2—FEB. '84 $4.00
Illustrated tour of the 1541! Artificial intelligence! Synapse's Ihor Wolosenko interviewed! And ready to enter: Music Maker Part II! Screen Manipulation! Night Attack! Relative Files!



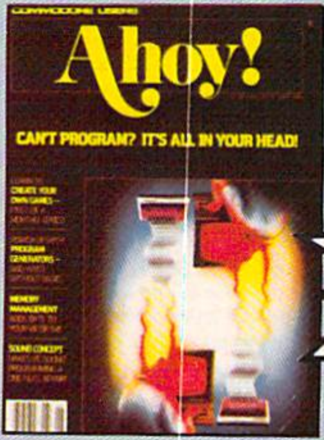
ISSUE #3—MAR. '84 $4.00
Anatomy of the 64! Printer interfacing for VIC & 64! Educational software series begins! And ready to enter: Address Book! Space Lanes! Random Files on the 64! Dynamic Power!



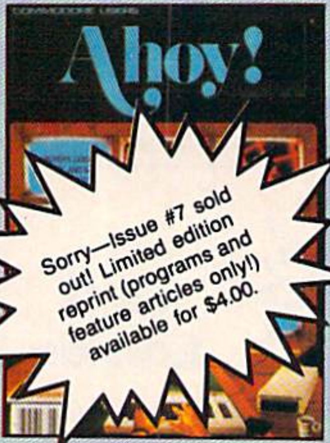
ISSUE #4—APR. '84 $4.00
Petspeed and Easy Script tutorials! Printer interfacing continues! Laserdisc! And ready to enter: Apple Pie! Lunar Lander! Name that Star! Lower case descenders on the 1525!



ISSUE #5—MAY '84 $4.00
Future of Commodore! Inside BASIC storage! Memory management on the VIC & 64! Spreadsheets! And ready to enter: Math Master! Air Assault! Biorhythms! VIC Calculator!



ISSUE #6—JUNE '84 $4.00
Game programming column begins! Program generators! Rupert on inputting! Memory management continues! And ready to enter: Post Time for the 64 & VIC! Alpiner! Sound Concept!



ISSUE #7—JULY '84 $4.00
MSD Dual Disk! Database buyer's guide! Training your cursor! Screen displays! File Sleuth! Users Groups! And ready to enter: Renumbering! Checklist! Math Defender! Brisk!



ISSUE #8—AUG. '84 $4.00
Choosing a word processor! Computational wizardry! Creating your own word games! Sound on the 64! And ready to enter: Micro-Minder! Directory Assistance! Terrible Twins!

ISSUE #9—SEPT. '84 $4.00
Program your own text adventure! Build a C-64 cassette interface! Video RAM! And ready to enter: Salvage Diver! DOS! Sound Explorer! Castle of Darkness! Base Conversions!

ISSUE #10—OCT. '84 $4.00
C-64 Graphics programs! Bit-mapped graphics! Joystick programming! And ready to enter: VIC 40 Operating System! BAM Read & Print! Emerald Elephant of Cipangu! Lawn Job!

ISSUE #11—NOV. '84 $4.00
Music programs & keyboards for the 64! Graphics feature continues! And ready to enter: PTE word processor! Block Editor! Alternate Character Set for the 64! Tunnel of Tomachon!

ISSUE #12—DEC. '84 $4.00
Buyer's guide to printers! 1525 printer tutorial! Custom characters! User Guide to KMMM Pascal! Diving into BASIC! And ready to enter: Construction Co.! Space Patrol! Cross Ref!

ISSUE #13—JAN. '85 $4.00
VIC/64 OS exposed! Sprites! 1541 device # disconnect switch! Ghostbusters! And ready to enter: Ultra Mail! Music Tutor! Alice in Adventureland! Midprint! To the Top!

ISSUE #14—FEB. '85 $4.00
Printer interfacing! Multicolor sprites! Modems! Bulletin boards! Theory of game design! And ready to enter: Futurewar! Fontasia! VIC Eraser! Insurance Agent! Telelink 64!

ISSUE #15—MAR. '85 $4.00
Making multiscreen gameboards! Inside the Plus/4! Commodore DOS! And ready to enter: Old Routine! Programmable Functions! Automatic Line Nos.! Home Budget! Salmon Run!

ISSUE #16—APR. '85 $4.00
Assembly language column begins! Programming the joystick! 1541 disk drive alternatives! And ready to enter: Hop Around! Faster 64! Booter! Elecheck! BASIC Trace! Space Hunt!

Send coupon or facsimile to:
***Ahoy!* Back Issues**
Ion International Inc.
45 West 34th Street—Suite 407
New York, NY 10001

If you have a modem and want more complete information on any of the issues listed, call the *Ahoy!* BBS: 212-564-7727 (as of April 15: 718-383-8909).

Ahoy!

Please Send Me The Following:

____ Copies of issue number ____
____ Copies of issue number ____
____ Copies of issue number ____

Enclosed Please Find My Check or Money Order for $________

(Outside the USA please add $1.00 for every copy)

NAME ________________
ADDRESS ________________
____
CITY ________________
STATE ________________
ZIP CODE ________________

WordPro was among the first office quality word processors for the C-64. Their latest entry, *Fleet System 2,* easily outclasses *WordPro*, for it packs an onboard spelling checker and several other distinctive features that are handy in an office environment.

Business users will appreciate the choice of viewing a 40-, 80-, or 120-column display. (The screen scrolls horizontally when you're keyboarding in the wider modes, which are excellent for preparing heavily tabulated reports.) A setup program that is run prior to the first time you use the word processor allows you to pick the colors of the screen's border and background as well as the text. Regrettably, the display lacks automatic word-wrap. Words get cut off at the right column and are continued on the next line, which restricts onscreen readability. There is one consolation. When working in 40-column mode, you can "preview" the document and see a vertically scrolling 80-column display of the document's ultimate appearance and the effects of all formatting commands. (You cannot write and edit while previewing, however.)

While composing this review with *Fleet System 2,* I was considerably dismayed to see some of my snappiest sentences abruptly vanish into the Twilight Zone. Whenever I pressed the return key while the cursor sat on an existing line of text, everything between the cursor and the right side of the screen disappeared. My spirits were soon lifted, however, by the discovery of a feature called the "extra text" area, which accommodates as many as 450 screen lines of text that can be manipulated in several practical ways. While learning the system, for example, you can load a "help" file into this area and flip back and forth between your ongoing document and a list of commands and functions and their respective keystrokes. Frequently used words, addresses, even whole chunks of text can be kept here and inserted into the ongoing document. This is accomplished by designating abbreviations for them in the extra text area. For this review, I reserved the letters "od" to indicate the words "ongoing document." While writing the above sentences, I merely typed the f1 key, the letter "a" (to indicate I wanted to "append" a phrase from the extra text section) and the letters "od". Text stored here can also be automatically inserted into blank space within a document, which serves to facilitate the rapid printing of personalized form letters.

When you're working in the main text area, a status line that looks like : X : F : I : S : N : reminds you of the current editing mode. The f1 or RUN/STOP key works like a clutch. Press f1 and the F lights ups; then hit X to enter the extra text area. A similar two- or three-keystroke process is also engaged to conduct operations such as marking a range of lines to be copied, erased or funneled off to another part of the ongoing document. A range is marked by moving the cursor down to highlight the chosen text, but you can only highlight whole lines. If only half the line is to be transferred or erased, you must use the "paragraph split" command to shove the other half down to the next line. This is the most awkward aspect of editing with *Fleet System 2.* Another common editing operation, shifting between insert and overwrite modes, is swiftly executed by tapping the f3 key.

Cursor control is limited to hopping to the top of the screen, beginning of the document, or down to the next line. Of course, you can glide freely around the current screen via the cursor keys, but cannot skip directly to the end of the immediate screen or the document, or flip to the next or previous screenload. You can "speed-scroll" through the text, but this still isn't as useful as the ability to flip screens and is also rough on the eyes. At least it is easy to delete the next word or sentence, in which case the program highlights the text and offers you a chance to bail out before consigning the offensive text to extinction. *Fleet System*'s formatting commands look familiar: one or more lines contain marks such as lm5 (left margin is set for five spaces), which can be embedded throughout a document to change centering, justification, and other formatting features. Underlining and other printer-supported features can be addressed, and you may even indicate Spanish accent marks if using a dot matrix printer. (Compatible printers are listed in the setup program's menu and encompass a range of twenty-three popular machines including the FX-80, TTX, Gemini, Brother, Okidata 82, TP-1, and the Commodore printers.)

Fleet System's major selling point, a 70,000 word dictionary, is on the opposite side of the program disk and can be accessed without exiting the word processor. It rapidly checks all the words against this dictionary and one created by the user, then highlights all "suspect" words. These may be ignored, added to the user dictionary, or corrected on the spot. (The user dictionary can hold up to 12,000 words.) After saving the document and exiting to BASIC, you may avail

yourself of some intriguing features in the options program. A frequency and statistics report will list all selected words and the number of times they appear. You'll also be informed of the number of different words, average word length, number of sentences, and other statistics that may or may not be significant to you. (You don't have to access this program for the most practical such option: two keystrokes will instantaneously summon up a precise word count at any time.)

A single document may be up to eight or nine pages in length, and files can be linked to simultaneously print a document that was broken into several files. Saving and loading files is a simple process, and the program supports one or two disk drives. Headers, footers, and page numbering, Search and Replace, mailing labels and other standard features are effectively implemented, and *Fleet System* has a couple of features that are less common. You can load sequential files saved by a database or other program, convenient for preparing the data to be used in form letters and other mail merge operations. Numeric tabs simplify aligning columns of numbers containing decimal points. While in numeric mode, you may add or subtract columns of numbers. You can also total all the numbers in a paragraph or range of lines, an unusual and creative option.

Fleet System is easy to learn and operate. The manual has a lucid tutorial for beginners; old hands will prefer to consult the index and flip to the appropriate page in the reference section. Because the program doesn't have enough cursor control "shortcuts" and makes range-setting a relatively awkward process, it is ill-suited for composing term papers, essays, or anything demanding intensive revision. *Fleet System 2*, with numerous features that emphasize business-oriented word processing, is more valuable in an office where the workload includes extensive preparation of tabulated and columnar material.

Professional Software, Inc., 51 Fremont Street, Needham, MA 02194 (phone: 617-444-5224).

–Shay Addams

SUPER PRINTER UTILITY PROGRAMS

Cardco, Inc.
Commodore 64
Disk; $39.95

The newest addition to Cardco's *Cardware* software line, this disk contains such nifty entries as screen dump utilities, a banner headliner, rudimentary word processing and mailing list programs, a calculator, a screen drawing program, and a sprint print printer buffer. Also included are file conversion, BASIC program cross reference, 24K file printer buffer, and index generator programs. On the flip side is a utility for viewing the screen images also found on this side of the disk.



Sample screen dumps. Ahoy! *logos generated with banner headline utility.*
READER SERVICE NO. 195

The *Super Printer Utilities* program, the most impressive of the lot, allows you to program into the C-64's memory specific data about the type of printer you're using. It also provides several utilities for executing screen dumps, initializing and exiting keyed routines, ASCII screen dumps, reverse screens, and a host of other useful POKEs and SYS commands for using the features of the program. Once loaded, the familiar "ready" prompt appears on the screen, telling you that this data is now in memory. One nice feature is that you can *specify* in which block of memory you'd like to store this printer data and command sequence. You're given a choice of $C400, $C000, $9400, $7400, or $5400. This is useful in avoiding "address conflicts" when programming in machine language. Once you've chosen the load address, the POKE values and SYS commands for that address are displayed. You're also given the option of printing this information out for future reference, if desired. For example, if $C400 were chosen as the load address, the corresponding command list would read:

Screen Dump (Hor.)	SYS 50176
Init Keyed Routines	SYS 50179
Exit Keyed Routines	SYS 50182
Screen Dump (ASCII)	SYS 50185
Screen Dump (Vert.)	SYS 50188
Reverse Screen	POKE 50194, S
Reverse Character	POKE 50195, C
Force Flag	POKE 50196, F
CK01 Enable	POKE 50197, KE
Output Mode	POKE 50198, OM
Device #	POKE 50199, DV
Secondary Address	POKE 50200, SA
Width (Hor.)	POKE 50201, W
Color Mode (Vert.)	POKE 50202, CM
User Screen Enable	POKE 50203, UE
Lores Start Page	POKE 50204, LO
Hires Start Page	POKE 50205, HI
Color Start Page	POKE 50206, CO
Prntr Type (ASCII)	POKE 50207, PT
Non-ASCII (ASCII)	POKE 50208, NA
2x2 Color Dots (Vert.)	POKE 50211, C2
4x2 Color Dots (Vert.)	POKE 50243, C4
Keypad Definitions	POKE 50275, KD
Utility Programs	50417

Examples of the screen dumps possible are shown herewith. Each of the image dumps was executed with a simple SYS command; one problem I encountered, however, was the frequent crash of the pro-

gram when trying to dump more than one screen. I was able to return to the selection menu sometimes—on most occasions, though, the screen would scroll with a "FORMULA TOO COMPLEX ERROR" message and I couldn't recover from the crash, no matter what I did. This meant I would have to shut off the C-64, turn it back on, and reboot the program from scratch. If it only happened once or twice, I wouldn't even mention it. But since it was the rule rather than the exception, be aware that this program has a definite tendency to self-destruct!

I've also included some dumps of the "BANNER HEADLINER" program contained on the disk. You're given a choice of "Gothic" or "magnetic" type styles, and a choice of short or tall letters. Once these choices have been made, you may enter up to 254 characters of text or punctuation and dump it to your printer. Possible uses for this feature include making banners (happy birthday, welcome home, etc.) or signs for flyers (garage sale today, no parking).

The other utilities are of the "bare bones" school. The word processor has no refinements whatsoever, nor does the mailing list utility. They are useful, however, for extremely light duties.

The 44-page instruction manual is good as far as it goes, but contains an awful lot of ambiguity. A case in point is the print-dump command listing above. Take a look at the last line: "Utility Programs....50417." Is that POKE 50417 or SYS 50417? While it may not seem like a big deal, if you need to access one of these features, typing in a SYS instead of a POKE may result in a crash. Another question unanswered: is it possible to dump, say, a playscreen from *Zaxxon*? It probably is, but *I* don't know how—the documentation fails to make any mention of such abilities.

Users of Cardco's *Write Now!* word processor will like the program however, since it includes utilities to convert ASCII to *Write Now!* format and vice versa. Additionally, it will allow you to convert *Write Now!* files for transmission over a modem.

All factors considered, I was underwhelmed by the *Super Printer Utility Programs.* The crashes I mentioned and the lack of complete documentation made the disk tedious to use, and not at all up to Cardco's usual fine standards.

Cardco, Inc., 300 S. Topeka, Wichita, KS 67202 (phone: 316-267-6525). *—Tom Benford*

STAR SG-10/15 PRINTERS
Star Micronics, Inc.
SG-10: $299; SG-15: $499

I've used a Star Gemini 15X dot matrix printer for the last two years, and come to regard it as a reliable workhorse, proven through thousands of hours of use to be rugged, dependable, and flexible for a great number of applications. So when Star announced its new, improved version of the 15X, called the SG-15, I knew I had to get my hands on one and put it through the paces.

What Star has done is make a good thing even better. All the features of the older models have been retained, but enhancements have been made. Even the throughput speed has been increased by a full 20%! The added features make the SG-10 and SG-15 the printers many users have been waiting for.

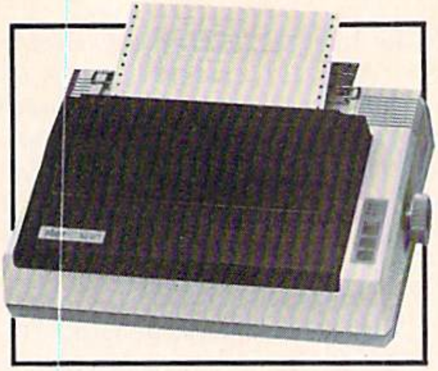
The SG-10 and SG-15 are direct replacements in the company's product line for the Gemini 10X and 15X. The difference between the two new models is the carriage size, the SG-10 having a 10" carriage, the SG-15 a 15". (The one other difference is the 15's 16K buffer—see below.) Since I often have occasion to use spreadsheet software and need the wide paper capacity, I chose the 15.

Externally, the SG-15 looks quite similar to its predecessor, the 15X. The function controls are of the "soft-touch" variety, replacing the push-switches of its older brother. That's the only obvious physical difference. The other improvements are more subtle, but they show their worth as soon as you activate the printer.

The Gemini 10X and 15X were equipped with 4 DIP switches inconveniently located on the back of the printers. These were used to select the resident character set on power-up, linefeed on or off, end of paper, and other printer functions. On the new SG-10 and SG-15, the DIP switches have been moved to a more accessible location: the left side of the printer. But in addition to making them easier to reach, 8 DIP switches have been added to the original 4. Why the extra switches? To handle the configurations for the enhanced functions, natch! And what enhanced functions they are.

The SG-15 comes with a 16K print buffer as standard equipment, double the capacity of the earlier Gemini 15X. This means that the printer itself can store up to 16K of information and hold it in memory until printed, freeing up the computer for other uses while printing continues. This humongous buffer proves its worth when using word processing software or printing out lengthy BASIC program listings. An optional buffer board is available for the 10.

Both SG models feature (as did



SG-10 & 15: near letter quality.
READER SERVICE NO. 201

their predecessors) friction and tractor feed as standard, allowing you to use tractor-feed paper for listing and/or rough drafts as well as single sheets for correspondence. These new models also allow hex dumps and downloadable character sets. On internal ROM are international character sets for French, German, English (Britain), Danish, Swedish, Italian, and Spanish. These sets contain umlauts, pound signs, circumflex letters, and other characters and punctuation indigenous to the specific language. Additionally, you may custom-design your own characters and load them into the printer, useful for including logos and other "personalized" touches in your printouts. Of course, the entire graphic character set is available for printing out listings of your programs exactly as they were entered. Both the SG-10 and SG-15 are capable of printing out ultra high resolution bit image graphics as well. Several varieties of condensed, expanded, and enhanced print are also possible on either of these models. Of course, the printer interface you utilize has a lot to do with how many "tricks" the printer

```
Normal Star mode.
Bold print sample.
Near Letter Quality.
Underscored sample
Italic sample.
Emphasized mode.
Expanded.
Condensed print mode.
Condensed with underscore.
Condensed bold print.
```

Print samples from SG-10 and 15.

will perform, but with the added DIP switches, it's easy to get the most out of just about any of the popular printer interfaces. Both the SG-10 and 15 come with Centronics parallel input, but either can be had with RS-232 serial input as an option.

A few words about the documentation are in order. The 235-page user manual is superb, providing information on everything you need (or want) to know about DIP switch settings, control characters and functions, changing defaults–the whole nine yards. The only thing I took exception to was the continual mention of "IBM mode." The printers have two basic operational modes: Star and IBM. Either is selected or deselected by the DIP switches. This is to accommodate the broadest possible market of users, and apparently the folks at Star figure these two printers to be hits with Big Blue users. Included in the manual are tables and pin-out assignments for Apple, TRS-80, and CP/M-based computers, but nary a word for us Commodore folk! Get with it, Star! Don't you realize that there are more Commodore 64s out there than the other brands *combined*?

A host of useful provisions are included in the printers' operating systems, such as the capability for superscripts, subscripts, variable horizontal spacing control, multistrikes for boldface, underlining, and italicizing. These printers are truly a dream to use for word processing with the inclusion of these useful and powerful features.

Both of the SG printers use inexpensive multipass two-spool nylon ribbons. I've always liked this feature about Star's printers, since these ribbons are cheap and readily available at most stationers.

I've saved the best and most impressive feature for last: near letter quality (NLQ) printing. As the name implies, this mode emulates the print image of a daisy wheel printer or traditional typewriter. It does this so well, in fact, that unless you look very close at the printout, you'll think it is the product of a daisy wheel Print speed is reduced by about half in this mode, since the print head

makes two passes for each character. (The first pass prints out the "normal" or draft-mode character, while the second pass fills in the open area between the dots and embellishes it somewhat, adding serifs and other finishing touches.) In the past, I've used the Gemini for my rough drafts and a daisy wheel for the finished product. The print quality is so good in the NLQ mode that I've become a "one printer user" again. Anybody want a used daisy wheel printer?

Star Micronics, Inc., Pan Am Building Suite 3510, 200 Park Ave., New York, NY 10166 (phone: 212-986-6770). *–Tom Benford*

TROLLS & TRIBULATIONS
Creative Software
Commodore 64
Disk; $24.95
(See screen on page 63)

Longtime joystick jolters often feel there's nothing new under the sun. Their problem is that they're looking in the wrong direction. A look under the street will reveal *Trolls and Tribulations*, Creative Software's sewer-based action game featuring superb music and graphics and a veritable cesspool of activity.

Your eight-lived troll must traverse a scrolling multilevel maze, shooting cretins and eschewing skulls, buzzards, and spiders while collecting treasure. Each move must be planned with care; blindly leaping from peak to precipice will send your troll plunging into the lapping waters below. That is a fate you want to reserve for the cretins that descend from above. Your troll's shots turn them into eggs which you attempt to shove to a watery grave before they hatch.

At the onset, you choose from novice, intermediate, and expert levels of play by opening the corresponding door. Your troll then enters the first set of horizontally scrolling mazes. Locked doors appear at the end of all sections, only to be opened by the key which you acquire within the maze. Keys and treasures are collected by leaping from ledge to ledge or dropping to lower shelves. Strategically placed trampolines will assist in reaching higher levels. Players may at first be inhibited by the need to vault moving skulls and buzzards while within low-ceilinged tunnels, but your troll cannot hit his head.

While the lower levels are quite masterable, the higher ones will prove a true challenge. The progression from easiest to hardest is gradual enough to maintain interest as ability increases.

I do have two bones to pick with an otherwise delightful game. First, there is no way to bypass earlier sections and begin deeper in the maze. Second, high score is provided, but no means of saving the information to disk (the instructions cite memory limitations). Who cares about keeping track of high score in a single sitting?

Creative Software, 230 East Caribbean Drive, Sunnyvale, CA 94089 (phone: 408-745-1655).

–Scott Kincaid

PFS:FILE
PFS:REPORT
Software Publishing Corporation
Commodore 64
Disk; File $79.95, Report $69.95

PFS:File, along with your C-64, disk drive, and optional printer, forms a fairly full-featured database management system. You design the record format with full use of cursor control, delete, and insert keys. You have the top 21 lines to designate as many as 50 fields. Not enough? Hit a key and you get a blank screen. More and more—up to 32 screens full. Be careful with those long record formats, though; if you fill out every field in a 32 screen record, only five records will fit on your diskette. Realistically, for a simple name and address file, you will be able to store about 500 records. If the addresses are short, you might even store *File's* maximum of 1000 records on your diskette. A "percent full" reading at the bottom of the screen will give you some idea of how you're doing.

From *File's* main menu, you can choose to design a new format or change an old one, add data, search through the records you have entered and update those with mistakes, print or delete records individually or in groups, or return to BASIC. As you work to create the perfect design for your records, use the Insert function cautiously. The Clear key is right next to the Insert key; hit Clear and all your design work will disappear. The designers of *File* should not have made clearing the screen so simple. When you design your second database, with a different record format, you will need to use a new diskette. if you have lots of short lists to keep, *File* can eat up your blank disks. And don't forget to format those blank data disks before you load *File;* it has no capability to format them for you.

When you want to search for a particular record, *File* can match as many fields as you specify. You can ask for an exact match (full first and last name), a partial match (a word or characters in the middle of an entry), a range of numbers (all zips from 32900 to 32950), or a "not" match (all records that do not match the search specifications). Searching for a record based on the contents of the first field will never take more than three seconds. Use any other field and the search may take much longer. *File* does not allow you to save your search specifications to disk.

Getting a printout of your records is easy. First tell *File* which records to print using the same options that are available for a search. Then tell it which fields to print from each record and whether to print each field on the same line or a new line. This option allows you to print mailing labels; however, since *File* can only print fields in order, the possible formats for your records are limited. *File* can sort as it prints, but it can only sort on one field. *File* lets you store up to eight print formats including search specifications that you can use or edit at a later time.

A critical feature of any database

is the ability to redesign you data files. Inevitably, you will want to add a field to each record, rearrange the fields, or delete an unnecessary field to make more room on the disk. *File* handles all these changes with no problem as long as your disk is less than 50 percent full. It is a pleasure not to have to retype dozens of records just because you have changed your mind about layout. *File* can only redesign four pages at a time, so a five page form will require two passes. And redesigning a form is slow, oh so slow.

One caveat–any new fields that are to retain their data must have labels identical to the old fields. *File* does something strange when you set up your labels (or field names); each one must end with a colon. When reorganizing a database, *File* matches every character between one colon and the next. If you have used a row of dashes to break your form into upper and lower halves, *File* will think those dashes are part of the label which follows them. This "feature" means you may want to be very careful (read: avoid) using a heading or any other text that is not actually a field name.

Records can be deleted individually during a search or in groups. Give *File* a set of search specifications and tell it to delete; your data will disappear and free disk space will reappear. *File* numbers your records consecutively as you enter them. When some records are deleted, the others are not renumbered. Only if you redesign the form will the numbering be consecutive again. But after a redesign the records are numbered in reverse order, so you wil have to redesign the database twice to have the latest records numbered highest.

File has no number crunching capability and only limited printing options. Get your wallet, here comes *PFS:Report*. *Report* takes your *File* database and lets you design and save up to eight report formats. A report organizes your fields into columns and prints them in spreadsheet format. You can choose up to 20 fields to be printed as columns. Based on the contents of the first and second columns, *Report* can sort your data in ascending alphabetical order (A to Z) or in descending numerical order (1000 to 0). *Report* has a very powerful feature that lets you sort data based on key words in any one field. This could really automate a research project; like redesigning a form, though, it is slow.

Whatever page title you specify will be printed on each page of the report. The heading for each column will be the label of the corresponding field unless you specify some other heading. *Report* can give you the total, count, or average of any numeric column. Additionally, it can calculate subtotals, subcounts, or subaverages each time the entry in column one changes. If you just want the bottom line, *Report* will print a summary with totals only.

If that's not enough, *Report* can also calculate across rows. Say your inventory list has entries for price paid and selling price. *Report* can generate a column that shows profit (based on those two entries) for each inventory item. Up to three such "derived" columns can be contained in any report. *Report,* unlike *File,* will let you print columns in any order; they do not have to be in the same order as the fields in your database.

The manuals for *File* (24 pages) and *Report* (63 pages) are in the nice 6½ by 8" wirebound format that lays flat on your desk. The chapters match the programs' menus, and each chapter combines a tutorial with reference information. Each manual contains a quick guide that you can use to refresh your memory once you have learned the program. Screen representations rather than actual pictures are used, so the illustrations are not identical to what you'll see on the monitor. All the programs' error messages are explained, but this is hardly necessary

Continued on page 84

COMMODARES

PROGRAMMING CHALLENGES

By Dale Rupert

Each month, we'll present several challenges designed to stimulate your synapses and toggle the bits in your cerebral random access memory. We invite you to send your solutions to:

Commodares, c/o *Ahoy!*
P.O. Box 723
Bethel, CT 06801

We will print and discuss the cleverest, simplest, shortest, most interesting and/or most unusual solutions. Be sure to identify the Name and Number of the problems you are solving. Also show sample runs if possible, where appropriate. Programs on diskette are welcome, but they must be accompanied by listings. Also tell what makes your solutions unique or interesting, if they are. You must enclose a stamped, self-addressed envelope if you want any of your materials returned.

Your original programming problems, suggestions, and ideas are equally welcome! The best ones will become *Commodares*.

Problem #17-1: Decimalizing Dates

This problem was suggested by John Immarino (Hackensack, NJ). Write a short program to convert the string form of a date into a numerical decimal value. For example, if the user enters 1/1/85, the computer returns 1985.0000. Since mid-year is around July 1, if the user enters 7/1/83, the computer returns 1983.5000. John points out that these decimal values would be easier for performing calculations involving time periods than the string versions are. He also reminds you to adjust for leap years!

Problem #17-2: Printer Sentinel

Here's a problem sent by R.W. Kober (Buffalo, TX). Write a subroutine to determine whether or not the printer is turned on. If the printer is on, the subroutine returns immediately. If the printer is off, the subroutine displays a message such as "The Printer Is Not On." The subroutine waits until the printer is turned on, at which time it returns to the main program. This should be a tidy, unimposing subroutine.

Problem #17-3: MID$ Statement

Some versions of BASIC contain a MID$ statement. Commodore BASIC allows only the MID$ function. The difference is that the MID$ statement (pronounced "mid-string") appears on the left side of the "=" sign in an assignment statement whereas the MID$ function appears on the right.

Your challenge is to write a subroutine to simulate the MID$ statement. The syntax is this: MID$(X$,S,N) = Y$. The result is that some or all of X$ is replaced by some or all of Y$. Specifically, the first N characters of Y$ are substituted for characters in X$ starting at position S of X$. The length of X$ must not change. The value of Y$ is unchanged. Some examples should help.

Initially X$="RUN" and Y$="GO". S and N are assigned the following values before your subroutine is called. The value of X$ upon returning from your subroutine is shown.

	N=0 or N>1	N=1
S=1	GON	GUN
S=2	RGO	RGN
S=3	RUG	RUG

N may range from 0 to 255, and S may range from 1 to the length of X$. If N is greater than or equal to the length of Y$ or if N is 0, then all of Y$ will be used (with the stipulation that the length of X$ will not change).

Problem #17-4: Number Speller

The user enters a number between one and one thousand, and the computer spells that value. For example, if the user enters 12, the computer displays "twelve." If the user enters 235, the computer returns "two hundred thirty five." Anyone submitting thousand-item DATA statements will be sent to the back of the class.

First this month, a correction to the centering function described in the January 1985 edition of *Commodares.* The correct definition for FNCE is DEF FNCE(X)=INT(20−X/2). Type X=LEN(A$) : PRINT TAB(FNCE(X)); A$ to print A$ centered on the current (40-character) line. Thanks to Milton Powell and Daniel Miller for pointing this out.

As promised last month, we have solutions to *Problem #16-2: Quick Decimal* from the originators of the problem. Programs from Jim Speers (Niles, MI) and John Prager (Bay City, MI) are listed below:

```
1 REM PROBLEM #16-2 : QUICK DECIMAL
2 REM SUBMITTED BY JIM SPEERS
3 REM
10 INPUTN$:FORI=1TO4:A=ASC(MID$(N$,I,1))
```

```
-48:T=T+16^(4-I)*(A+7*(A>9)):NEXT:PRINTT $

1 REM PROBLEM #16-2 : QUICK DECIMAL
2 REM SUBMITTED BY JOHN PRAGER
3 REM
10 INPUT H$:N=0:L=LEN(H$):FOR J=1 TO L:A
$=MID$(H$,J,1)
20 K=VAL(A$)-(A$>"9")*(ASC(A$)-55):N=N+K
*16^(L-J):NEXT:PRINT N
```

Jim's solution has the advantage of being a one-liner, but four hex digits must always be entered. John's program allows hexadecimal values of any length to be entered, at the expense of an additional program line.

By far the most popular *Commodare* in January was *Problem #13-1: Starred String.* The two most general categories of solutions are a) screen and printer compatible and b) only screen compatible. Programs which used cursor movements are of course not usable for direct printer output. The award for most unusual solution goes to Michael Hommer (Waukee, IA) for his program written in COMAL. Here's what it looks like:

```
INPUT AT 2,4,35: "type in the string : "
; instr$
star'string(instr$)

// Procedure

PROC star'string(instr$)
  DIM star$ OF 40, string$ OF 40
  string$:="* "+instr$+" *"
  star$:=""
  FOR i:=1 TO LEN(string$) DO star$:+"*"
  PRINT star$
  PRINT string$
  PRINT star$
  ENDPROC star'string
```

The first line prompts the user for an input. The second line executes the "star string" procedure which is defined in the following lines. The FOR-DO statement does not extend past the end of its line. The program logic should be easily understandable with a little study even for those who are not familiar with the language. Interesting solution, Michael.

Examples typical of most other submissions are listed below:

```
1 REM PROBLEM #13-1 : STARRED STRING (SC
REEN ONLY)
10 INPUT A$:PRINT"[DOWN]* "A$" *":FOR X=
1 TO LEN(A$)+4:PRINT"[UP][UP]*[DOWN][DOW
N][LEFT]*";:NEXT

1 REM PROBLEM #13-1 : STARRED STRING (PR
INTER/SCREEN)
10 INPUT A$:FOR X=1 TO LEN(A$)+4:X$=X$+"
*":NEXT:PRINT X$:PRINT"* "A$" *":PRINT X
$
```

Both of these programs are one-liners if you use abbreviations and omit spaces. Variations on these themes involved using MID$ and LEFT$, and some programs called a star-printing subroutine. To give credit where it's due, here is a pseudo-randomly arranged list of readers with workable solutions to this and some of the other problems:

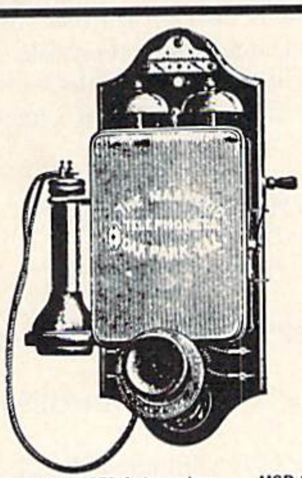
Jimmy Toten (Hawthorne, CA)
R.W. Kober (Buffalo, TX)
Milton Powell (Portsmouth, NH)
Daniel Miller (New York, NY)
Kenneth Karow (Chicago, IL)
Eric Biberhofer (Dundas, ONT)
Dave Patterson (San Jose, CA)
Heike Hamacher (Tulsa, OK)
Dan Berger (Marshfield, WI)
Fred Randall (Uxbridge, MA)
Stephen Vanderhoof (Albany, NY)
Mark Harris (Blue Springs, MO)
Jim Root (Whimore Lake, MI)
Richard Bertke (Perryville, MO)
David Wright (New Britain, CT)
Arnie Friedman (Great Neck, NY)
Garrett Kearney (Clay, NY)
Alan Christman (Romeo, MI)
Bill Imrisek (Bancroft, ONT)
Brian Wilcox (Coldwater, OH)
Shawn K. Smith (Bronx, NY)
Tim Stonis (Cranbury, NY)
Mike Bouska (Williamsburg, KS)
Greg Smisek (Lonsdale, MN)
Marcus Cooper, Jr. (Mililani, HI)
John Immarino (Hackensack, NJ)
Greg Sheppard (Tacoma, WA)
Michael Hooper (Tyler, TX)
Harold Hughes (Robstown, TX)
Ronald Sipe (Cincinnati, OH)
John Bucy (Belpre, OH)
Douglas Gristina (Florence, AL)
Barry Friedman (No. Miami, FL)
Steve Liggett (College Park, GA)
Kurt Huebner (Anchorage, AK)
John Prager (Bay City, MI)
James E. Borden (Carlisle, PA)
Todd Growney (Livingston, MT)
Thomson Fung (San Diego, CA)
Jon Phillips (Sierra Vista, AZ)
Charles Foreman (Dayton, TX)
Adolpho Rodriguez (Newark, NJ)
Robert Rice (Oak Park, MI)
Doug Chapman (Belleville, KS)
David Handwerker (Davison, MI)
Jeff Lambert (Honesdale, PA)

Your letters indicate you enjoy solving the *Commodares*. Occasionally readers ask for more challenging problems. Our goal is to present problems which can be solved with short to moderate-length programs for people with a wide range of interests and abilities. As always, if you have a problem that you have solved or would like solved, submit it so that we might dare the rest of the world to try it. *Commodares* will be as challenging as you make them.

Problem #13-2: Auto Run brought some hints, tips, and solutions from R.W. Kober (Buffalo, TX), James E. Borden (Carlisle, PA), Antoine Alary (Montreal, QUE), Todd Growney (Livingston, MT), Jim Zdunic (East Leroy, MI), Richard M. Bertke (Perryville, MO), and William H. Fossett of Hacker's Hardware (P.O. Box 7933, San Diego, CA 92107).

Jim Zdunic's program listed below is used for autoloading the next program from tape. After typing and saving his program, you activate it by typing RUN, NEW, and then SYS 7669. From then on, the next tape program will be automatically loaded and run whenever you type SYS 7669.

```
1 REM PROBLEM #13-2:AUTO RUN (FOR TAPE)
2 REM SUBMITTED BY JIM ZDUNIC
3 REM
10 FOR A=7669 TO 7679:READ B:POKEA,B:NEX
T:END
20 DATA 169,1,141,198,0,169,131,141,119,
2,64
```



Call us the modern way—

on *Ahoy!*'s Bulletin Board System!

If your computer is equipped with a modem, you can call *Ahoy!*'s Bulletin Board System any hour of the day, any day of the week to exchange electronic mail with other Commodore users or download files like the following:

- Editorial calendar for upcoming issues
- Excerpts from future editions of *Scuttlebutt*
- Corrections to programs and articles
- Detailed descriptions of back issues
- Program disk and subscription information
- Classified advertising

Set your modem for 300 baud, full duplex, no parity, 1 stop bit, 7 word length, and dial away!

System Configuration of *Ahoy!* Bulletin Board System:

C-64 and 1650 Automodem
Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380
Phone: 215-431-9100

CSS Monitor Stand
Computer Support Services
P.O. Box 927
Santa Barbara, CA 93102
Phone: 619-968-3040

MSD SD-2 Dual Disk Drive
MSD Systems, Inc.
10031 Monroe Street—Ste. 206
Dallas, TX 75229
Phone: 214-357-4434

Superstand I
A World of Plastic, Ltd.
2535 United Lane
Elk Grove Village, IL 60007
Phone: 312-860-2525

Impact Printer
Fidelity Electronics, Ltd.
8800 N.W. 36th Street
Miami, FL 33178
Phone: 305-594-1000

Uni-Kool C-100
Uni-Kool
909 Williamson Loop
Grants Pass, OR 97526
Phone: 503-476-1660

Vortex I BBS Software
Event Horizon Corporation
Box 1327, New York, NY 10028
Phone: 212-535-0697 (voice)
212-988-4539 (data)

Electra Computer Pad
Charleswater Products, Inc.
93 Border Street
West Newton, MA 02165
Phone: 617-964-8370

Sakata SC-100 Color Monitor
Sakata U.S.A. Corporation
651 Bonnie Lane
Elk Grove Village, IL 60007
Phone: 312-593-3211

Computer Stands #5933 & #0640
Royal Seating Corporation
P.O. Box 753
Cameron, TX 76520
Phone: 817-697-6421

NEW NUMBER (AS OF APRIL 15):
718-383-8909

If you use the Wedge with your disk system, you may autorun a BASIC program by typing "[UP ARROW] filename" where [UP ARROW] is the key next to the Restore key. James Borden and Antoine Alary pointed out that you may type the following to autorun a BASIC program without the Wedge:

```
LOAD "[FILENAME]",8 : [SHIFT-RUN/STOP]
```

The program below from Hacker's Hardware is the most versatile and easiest to use for creating a BASIC autoloader file. The user specifies the loader's name (for example, LDR) and the name of the BASIC program to be loaded (for example, TEST). Then the program creates the file LDR and puts it on the same disk as TEST. To autorun TEST, enter LOAD "LDR",8,1 or using the Wedge, simply enter %LDR.

```
5 REM   -- CREATE AUTO-RUN LOADER --
10 REM  COURTESY OF HACKER'S HARDWARE
20 INPUT"LOADER NAME";LN$
30 INPUT"NAME OF PROGRAM TO LOAD";B$
40 IF LEN(B$)>16 THEN END
50 IF LEN(B$)=16 THEN 70
60 B$=B$+CHR$(160):GOTO 50
70 OPEN 1,8,2,LN$+",P,W"
80 READ X%:IF X%=256 THEN 100
90 PRINT#1, CHR$(X%);:GOTO 80
100 PRINT#1,B$;:IF B$=""THEN CLOSE1:END
110 B$="":GOTO 80
120 DATA 187,2,169,15,166,186
130 DATA 160,0,32,186,255,32
140 DATA 192,255,169,16,162,240
150 DATA 160,2,32,189,255,169
160 DATA 0,32,144,255,32,213
170 DATA 255,134,45,132,46,169
180 DATA 15,32,195,255,32,83
190 DATA 228,32,89,166,32,51
200 DATA 165,32,142,166,76,174
210 DATA 167,256,139,227,187,2,256
```

Mr. Fossett invites readers to contact him at the address above if you are interested in an explanation as to how the program works. Several readers did some serious work on this problem.

There were basically two groups of solutions to *Problem #13-3: Micro Calc*, those that POKE 631 and those that don't. (The 631 refers to the first location of the keyboard buffer, for those of you who haven't memorized the memory map of the C-64.) First of all, some of you may have wondered, as did Michael Hooper (Tyler, TX), why would anyone want a program like this when the computer already performs calculations very simply in the immediate mode? The answer is that this was intended to be a parsing problem, a task frequently encountered in writing compilers and computer languages. Although it took some serious work and some clever programming to create some of the keyboard "buffer-stuffer" solutions, those programs don't provide much insight into the parsing problem. Certainly no one would type or load a program to accomplish a task they could do without a program.

The shortest example of a solution using the keyboard buffer was from Michael Hommer (Waukee, IA). The essence of this type of program is to POKE keystroke ASCII values into the keyboard buffer which begins at location 631. The number of keystrokes waiting in the buffer is POKEd into location 198. Once the BASIC program is run and returns to immediate mode, the computer reads the keystroke data from the buffer, just as it would if you had pressed those keys.

```
1 REM PROBLEM #13-3:MICRO CALC
2 REM SUBMITTED BY MICHAEL HOMMER
3 REM
5 INPUTA$:A$="?"+A$:PRINT"=":PRINTA$:M=6
31:C=145:POKEM,C:POKEM+1,C:POKEM+2,C:POK
EM+3,13:POKE198,4
```

The 145's correspond to [CURSOR UP] keystrokes and 13 is the ASCII value of the [RETURN] key. Other readers' solutions of this type included fancy color changes to hide intermediate output on the screen. If you haven't tried programming like this, you're in for some fun and interesting possibilities. Combine these keyboard buffer ideas with the autorun programs above, and your computer could get along without you for days at a time.

Parsing is the process of breaking the input sequence into its components and identifying or arranging them so that the desired operations can be performed. Shawn K. Smith (Bronx, NY) and Greg Smisek (Lonsdale, MN) sent similar solutions as listed below. Theirs were the most compact programs. Other readers took somewhat the same approach.

```
1 REM PROBLEM #13-3:MICRO CALC
2 REM SUGGESTED BY GREG SMISEK
3 REM  AND ALSO BY SHAWN K. SMITH
4 REM
10 INPUT"EXPRESSION";C$:FORX=1TOLEN(C$):
X$=MID$(C$,X,1)
20 IF X$="+" OR X$="-" THEN C=C+VAL(N$):
N$=""
30 N$=N$+X$:NEXT:PRINT"THE ANSWER IS ";C
+VAL(N$):PRINT:RUN
```

James Dunavant (Gainesville, FL) converted the keystrokes into their BASIC tokens and POKEd them directly into his BASIC program following a PRINT statement. Perhaps the most expandable approach is this one from John Prager (Bay City, MI).

```
1 REM PROBLEM #13-3:MICRO CALC
2 REM SUBMITTED BY JOHN R. PRAGER
3 REM
30 PN=1
```

```
40 GET A$:IF A$="" THEN 40
50 IF A$>="0" AND A$<="9" THEN N=N*10+VA
L(A$):CR=1:PRINT A$;:GOTO 40
60 IF A$="-" AND CR THEN GOSUB 120:PN=-1
:GOTO 40
70 IF A$="-" THEN PRINT A$;:PN=-PN
80 IF A$="+" AND CR THEN GOSUB 120:PN=1:
GOTO 40
90 IF A$="+" THEN PRINT A$;:GOTO 40
100 IF A$<>CHR$(13) THEN 40
110 GOSUB 120:PRINT"THE ANSWER IS ";S;".
":PRINT:RUN
120 PRINT A$;:S=S+N*PN:PN=1:N=0:CR=0:RET
URN
```

John's solution properly handles an input such as "−2". If you want to test the flexibility of your approach to this problem, see how easily you could modify your program to properly handle multiplication as well as addition and subtraction.

Problem #13-4: Data Lister was also a good parsing problem. The solution from Fred Randall (Uxbridge, MA) is listed below. His program is typical of several other readers' approaches to this one.

```
10 REM PROBLEM #13-4: DATA LISTER
20 REM SOLUTION BY FRED RANDALL
25 REM
30 N=20:DIM D$(N)
40 FOR J=1 TO 20:D$(J)=CHR$(64+J):NEXT
50 INPUT"+VIEW WHICH DATA";V$
60 L=LEN(V$):X=VAL(V$):X$=STR$(X):Y=LEN(
X$)+1:Y$=MID$(V$,Y,L):Z=VAL(Y$)
70 IF L=LEN(X$)-1 THEN PRINT D$(X):END
80 IF Z=0 THEN Z=N
90 IF X<0 THEN Z=ABS(X):X=0
100 FOR I=X TO Z:PRINT D$(I):NEXT
```

We have room for one final micro-challenge. When are spaces mandatory for proper syntax in a BASIC program? Never, you say? Think about it. See you next month with a new round. □

MOVING THROUGH *Color Memory*

Continued from page 22

string you want PRINTed, converts the ASCII characters to screen display codes, goes to screen memory, finds the current cursor position, and stores each character code in the correct address. It does it all automatically, far faster than you could do it by POKEing the screen codes into memory yourself.

But, as you know, part of that string can include "color" characters. For instance, CHR$(5) is the code for the color white. If you PRINT CHR$(5), every character PRINTed after that will appear white.

Yet there is no screen display code for "white," just as there is no screen display code for any of the other command characters, like carriage returns, cursor movements, or inverse on-off. These command characters are like a mini-language, telling the PRINT command what you want done with subsequent strings.

When you use a color command character, PRINT takes it as an instruction to PRINT all subsequent characters in that color.

But where is color controlled? It can't be controlled in screen memory, since there are 256 possible characters (include inverse characters) that can be coded. All eight bits of each address of screen memory are used just to determine which character will be displayed—there is no bit left over for color.

Instead, color has its own map, which corresponds exactly with screen memory. In the 64, color memory starts at location 55296 and includes all 1000 bytes through location 56295. In the VIC with less than 8K, color memory starts at 38400 and extends through 38905. (Expanded VICs find color memory at 37888 through 38393.)

When you PRINT a red Q in the upper left-hand corner of the screen, with the command PRINT CHR$(31) "Q", BASIC goes to the first address in screen memory and stores the code for Q. Then it goes to the first address in color memory and stores the code for *red*.

This is why the 64 can have a different foreground color for every position on the screen. Any position can be displayed in any of the 16 available colors, because each screen position has a character code in screen memory and a color code in color memory.

To see how this works, try this simple program. Line 10 fills the screen with random characters; line 20 fills color memory with random colors. Here is the 64 version:

```
10 SM=1024:FOR I=SM TO SM+999:POKE I,INT
(256*RND(9)):NEXT I
20 CM=55296:FOR I=CM TO CM+999:POKE I,IN
T(16*RND(9)):NEXT
30 GOTO 20
```

And here is the unexpanded VIC version (for expanded VICs, change the value of SM to 4096 and CM to 37888):

```
10 SM=7680:FOR I=SM TO SM+505:POKE I,INT
(256*RND(9)):NEXT I
20 CM=38400:FOR I=CM TO CM+505:POKE I,IN
T(16*RND(9)):NEXT
30 GOTO 20
```

Notice that because we POKEd the characters into screen memory, we could use the lower right-hand corner of the screen. The PRINT command automatically scrolls the screen display when a character is PRINTed in that position, but POKE, slow as it is, has the virtue

of leaving the rest of the screen alone when it puts a character there.

Now, you can easily see how we can accomplish highlighting certain positions on the screen *without* disturbing what is PRINTed there. We simply change the colors of that row in color memory, and regardless of what characters are coded to appear on that row in screen memory, they will have the color we assigned them in color memory.

That is how the example program *Menu* works. The options are PRINTed on rows 5, 6, 7, 8, and 9, all starting at the fourth character from the left. No option is longer than 16 characters. So the movement routine tracks five different addresses in color memory, which correspond to the starting positions of the five options. These addresses are held in the array C0*(n)*.

When the player calls for a change in the current choice displayed, the variable NP is set to the number of the new position, and the variable XP is set to the number of the old position.

To highlight an option, the routine at line 900 POKEs the highlight color code into the sixteen color memory addresses starting at the address C0(NP). To put an option back to normal, the routine at line 920 POKEs the regular color code into the sixteen color memory addresses starting at C0(XP).

Movement up and down responds to the f7 and f5 keys; the space bar is used to select.

The virtue of this program is that if you were changing menus, the same routines could be used to change colors in certain positions on the screen regardless of what characters were displayed there.

You will also notice that RUN/STOP and RESTORE are disabled at the beginning of the program. However, the program checks to see if RUN/STOP has been pressed—if it has, everything is set back to normal and the program ends. This is an example of how to keep the idiots savant from trashing the display. In this program, of course, it doesn't matter—but in the next one it does.

COLOR CODES

The color codes that you must POKE into color memory are identical to the color codes you POKE into the background color register to change the background color:

0 = black	8 = orange
1 = white	9 = brown
2 = red	10 = light red
3 = cyan	11 = dark grey
4 = purple	12 = medium grey
5 = green	13 = light green
6 = blue	14 = light blue
7 = yellow	15 = light grey

By the way, you can't just PEEK into color memory to read what color is stored there. That's because only the lower four bits of each color memory address location are under control. The upper four bits contain garbage. So to read color memory accurately, you must perform the operation AND 15 with the number you find there to get rid of the high nybble (the highest four bits), like this:

```
A=PEEK(CM+53)AND 15
```

After that statement, the variable A would contain the code for the color displayed in the 54th byte of color memory. (Remember, the first byte is CM+0, so that CM+53 is the 54th bute.)

FAST HIGHLIGHTING THRU FAKERY

The program *Menu* is slow. POKEing sixteen positions in color memory takes time, and the response feels sluggish.

But there's an easy way to speed things up on the 64. We can use PRINT to change color memory *without* changing screen memsory—by fooling BASIC into thinking screen memory is somewhere else.

You may remember from earlier jaunts through the video memory system that there is one location that tells the VIC chip where screen memory is, and another location that tells BASIC where screen memory is. BASIC uses its number when it is PRINTing character codes into screen memory; the VIC chip uses its number when it is scanning to see what patterns to display on the screen.

That means that if we tell BASIC that screen memory is in one place, and tell the VIC chip that it's in another, BASIC can PRINT all day and the VIC chip will never see it—it will still be looking in its own place for screen memory.

However, while BASIC is PRINTing invisible characters, all its *color* changes will be visible, because both BASIC and the VIC chip will be using the same color memory.

The example program *Fast Menu* shows how this is done. The options are PRINTed in screen memory at 1024, just as usual. But then the BASIC screen memory pointer at 648 is changed from 4 to 60. This tells BASIC to PRINT things at page 60, or address 15360 (that is, 60*256).

The program then PRINTs one string, XC$, to set a row back to normal, and another string, HC$, to highlight a row. It does a whole row (except the last character) each time, so that the option could be of any length.

The strings consist of the color command character plus 39 commas. They didn't have to be commas. They could have been anything, including blanks. What matters is that color memory is changed—we don't care what

Continued on page 114

TILE TIME FOR THE C-64

By Larry Murdock

Tile puzzles, the predecessors of the more recently popular cube games, were little plastic boards on which were laid several small interlocking tiles. Each tile or square had a number, a picture, or a letter of the alphabet printed on its face. When you received the game it was in order. You subsequently moved the pieces around to mix up the puzzle, then attempted to put it back in order.

Tile Time is just such a puzzle. Type it in and SAVE it.

When RUN, the puzzle will appear and a message at the bottom will read "JOYSTICK OR KEYBOARD J/K". If you select "K" the "cursor" can be positioned from the keyboard; up with an "A", down with a "Z", left with a "J", and right with a "." (see lines *7000* to *7080*). Hit the spacebar and the tile or tiles will move. If you wish to use the joystick, plug it into port number two (see lines *2000* to *2130*). If you want to move one tile, position the cursor on the tile you want moved and hit the fire button or spacebar. If you want to move more than one tile, position the cursor on the last tile you want moved and hit the fire button or spacebar. If you want to start all over, hit SHIFT and CLR/HOME simultaneously.

After you respond to the "JOYSTICK" question, the screen will read "MIX Y/N". If you respond with a "Y" the puzzle will mix itself with 50 quick random moves (you can halt the mixing process at any time, by hitting any key—see lines *4000* to *4200*). If you respond with a "N" then you can "MIX" it up yourself.

The code is kind of interesting and you can have some fun changing it around. The machine code scrolling is POKEd in via DATA statements (see lines *9000* to *49790*). You can change the face of the puzzle to anything you want (see lines *290* to *310*). The colors of the rows can be changed by altering the number codes in line 110. For younger players you can change one whole row to "A" 's and another to "B" 's, etc. If you don't speak well English you can change all the letters to numbers. If you confuse easily you can change every tile to the same thing so that no matter how mixed the tiles are, they will be in the right order. ☐

SEE PROGRAM LISTING ON PAGE 111

COMMODORE FONT EDITOR FOR THE C-64

BY PHILLIP CASE

Have you ever wondered how your computer knows to display the letter "A" when it's told to—especially when all it really understands is 1's and 0's? The answer lies in something called the Video Character Generator or Character Matrix.

Simply put, this is a Read Only Memory (ROM) which is filled with data that tells the computer what each character in the alphabet is supposed to look like.

Whenever the computer needs to display an "S," for example, it looks through the ROM until it finds the correct data for displaying an "S" on the screen. And since this matrix of data is stored in ROM, it's ready as soon as you turn on your Commodore 64.

Suppose you could change the data in the character matrix so the computer would display your own character set instead of the original. You could easily make your screen characters appear in Old English, or even other languages such as Russian or Greek, complete with their special letters. You could define special characters such as smiling faces, Pac-man, stars, etc. Imagine the impact of a game with a special character font to match the game theme. All this can be done.

The ROM which stores this character matrix cannot be altered, but there is a vector in the Commodore 64 which tells the computer where the ROM is. This vector is in RAM, which means it can be changed.

Therefore, if we change the vector so it looks into RAM to get the character matrix, we can have our own character information right there waiting for it. The purpose of the *Commodore Font Editor* is to provide an easy interface to create character fonts. The program itself is straightforward with few tricks. Type the program in slowly and try to understand each module as it is entered.

Before trying to design a character font, first develop it on paper. This allows you to verify the consistency between letters before you go into the editor. First, make up a character matrix such as the one pictured here. Then



define your character within the box. Remember to leave one side column and either the top or bottom row empty so your characters won't touch each other on the screen. If, of course, you are defining a cursive font, you will want the characters to connect with some sort of standard so they look consistent.

Also, you should not leave blank columns if you are designing special symbols that require more than one character, like a pointing hand.

USING THE FONT EDITOR

The first thing the program does after you type "RUN" is to load the character area with the data that is stored in ROM. This takes about 30 seconds and is only done the first time you type "RUN." If you stop the program and type "RUN" a second time, the program will not get the ROM characters. This is designed to prevent the program from wiping out a font you've been designing. The program checks for this by looking at the top row of the "@" character (Screen code 0), so don't alter the top row of this character.

After initializing the system, the program presents the main menu. From this point, you can go anywhere in the editor.

Note: the character codes displayed and inputs requested are not ASCII, but rather the Commodore Screen Codes, given on page 132 of the *C-64 User's Guide* and page 376 of the *C-64 Programmer's Reference Guide.*

Also, the character buffer for the font editor is from 12288 to 14335 (3000H to 37FFH). The program and variable areas from 2048 to 12287 (800H to 3000H). The program protects the character buffer from BASIC so your characters will not get wiped out by accident. ☐

SEE PROGRAM LISTING ON PAGE 108

Continued from page 12

trol of your computer to you in case of lockup. Four RENEW programs will restore the BASIC program in memory before throwing the switch. The unit's two female serial ports permit the use of two printers, and its two-foot cord acts as an extension to your serial bus.

$24.95 postpaid from Master Software, 6 Hillery Court, Randallstown, MD 21133 (phone: 301-922-2962).

BLUE CHIPS

The *Commodore Investor's Newsletter* is offered free to Commodore owners interested in playing the stock market. A copy can be obtained by sending a stamped and self-addressed business size envelope to Dr. Lynn Harner, 2011 Central, Dodge City, KS 67801 (phone: 316-227-8952).

APPLICATION TEMPLATES

Fabtronics has produced application templates for Microsoft's Multiplan for the Commodore 64. Current templates, priced at $11.95 for documentation and software ($6.95 for documentation alone), include home budget, accounts payable, accounts receivable, inventory management, check ledger, and invoicing.

Fabtronics, 51 Quarry Street, Brockport, NY 14420 (phone: 716-637-6371).

Cooperative Design is looking for individuals to write application overlays (to be called Magnetic Templates) for various C-64 spreadsheet and database programs. They will encourage users to modify, copy, and distribute these programs, with each new user voluntarily mailing in payment. Authors will receive 25 percent of all sales.

Send a stamped and self-addressed envelope to Cooperative Design at 129 Anthes, Box 138, Langley, WA 98260 and you'll receive a catalog of available and coming templates. If you'd like submission guidelines, send a list of templates you could develop and templates you'd like to see.

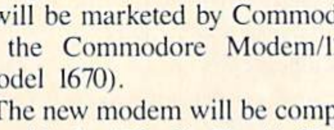
Also available: blank cardstock Personal RAM boards which fit around the keys of your 64 or VIC, providing space for notes for particular programs. Package of 12: $9.95.

1200 BAUD MODEM

U.S. Robotics has signed an agreement with Commodore International to manufacture a 1200/300 bit-per-second modem with automatic phone dialing and other advanced features. It will be marketed by Commodore as the Commodore Modem/1200 (model 1670).

The new modem will be compatible with the 128, 64, Plus/4, SX-64, and VIC 20 computers. As well, it will feature Hayes compatibility and a built-in speaker that will allow users to hear the call being dialed.

U.S. Robotics, Inc., 8100 McCormick Blvd., Skokie, IL 60076 (phone: 312-982-5010).



READER SERVICE INDEX

The publisher cannot assume responsibility for errors in the above listing.

ERRATA

Please note the following omissions/errors in previously published *Ahoy!* programs:

MANSION DISPLAY SETUP (March 1985):

The following line did not appear in print. Please insert it:

```
52723 DATA "[s U]X"
```

Some of you may have been confused by the extra open bracket in lines 52404 to 52408, and the extra close bracket in lines 52511 to 52517. These extra brackets appear not by mistake, but as characters which are printed to the screen.

OLD ROUTINE (March 1985):

We neglected to mention that the first byte of the program is 8000 and the last byte is 808C.

HOME BUDGET (March 1985):

Line 380, mangled during printing, should read as follows:

```
380 PRINT"[DOWN][DOWN]MONTHLY TOTAL"TAB(
14)Q+AA+BB
```

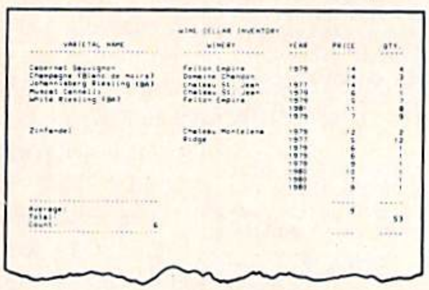
We regret any inconvenience these errors may have caused. Corrections to *Ahoy!* programs and articles are posted on our bulletin board (718-383-8909 as of April 15) as soon as they are spotted.

REVIEWS

Continued from page 74

since all the messages are in plain English anyway.

Together *PFS:File* and *PFS:Report* make a good team.



PFS: Report ***adds number-crunching and printer options to* File. *READER SERVICE NO. 236***

They give you all the important database features and *File* throws in a powerful extra. Your format is perfect, you're entering data; but you would really like to attach a note to one record. Just call up another page (for that one record only) and type your note. As simple as that. Just as the description of this feature is tacked onto the end of this review, *File* can attach a note to any record—and use a minimum of disk space to boot.

Software Publishing Corporation, 1901 Landings Drive, Mountain View, CA 94043 (phone: 415-962-8910).

—Richard Herring

PROGRAM LISTINGS

Attention new *Ahoy!* readers! You must read the following information very carefully prior to typing in programs listed in *Ahoy!* Certain Commodore characters, commands, and strings of characters and commands will appear in a special format. Follow the instructions and listing guide on this page.

On the following pages you'll find several programs that you can enter on your Commodore computer. But before doing so, read this entire page carefully.

To insure clear reproductions, *Ahoy!*'s program listings are generated on a daisy wheel printer, incapable of printing the commands and graphic characters used in Commodore programs. These are therefore represented by various codes enclosed in brackets [ ]. For example: the SHIFT CLR/HOME command is represented onscreen by a heart. The code we use in our listings is [CLEAR]. The chart below lists all such codes which you'll encounter in our listings, except for one other special case.

The other special case is the COMMODORE and SHIFT characters. On the front of most keys are two symbols. The symbol on the left is obtained by pressing that key while holding down the COMMODORE key; the symbol on the right, by pressing that key while holding down the SHIFT key. COMMODORE and SHIFT characters are represented in our listings by a lower-case "s" or "c" followed by the symbol of the key you must hit. COMMODORE J, for example, is represented by [c J], and SHIFT J by [s J].

Additionally, any character that occurs more than two times in a row will be displayed by a coded listing. For example, [3 "[LEFT]"] would be 3 CuRSoR left commands in a row, [5 "[s EP]"] would be 5 SHIFTed English Pounds, and so on. Multiple blank spaces will be noted in similar fashion: 22 spaces, for example, as [22 " "].

Sometimes you'll find a program line that's too long for the computer to accept (C-64 lines are a maximum of 80 characters, or 2 screen lines, long; VIC 20 lines, a maximum of 88 characters, or 4 screen lines). To enter these lines, refer to the *BASIC Command Abbreviations Appendix* in your User Manual.

On the next page you'll find our *Bug Repellent* programs for the VIC 20 and C-64. The version appropriate for your machine will help you proofread our programs after you type them. (Please note: the *Bug Repellent* line codes that follow each program line, in the whited-out area, should *not* be typed in. See the instructions preceding each program.) □

When You See	It Means	You Type		You Will See
[CLEAR]	Screen Clear	SHIFT	CLR/HOME	
[HOME]	Home		CLR/HOME	
[UP]	Cursor Up	SHIFT	↑ CRSR ↓	
[DOWN]	Cursor Down		↑ CRSR ↓	
[LEFT]	Cursor Left	SHIFT	←CRSR→	
[RIGHT]	Cursor Right		←CRSR→	
[SS]	Shifted Space	SHIFT	Space	
[INSERT]	Insert	SHIFT	INST/DEL	
[DEL]	Delete		INST/DEL	
[RVSON]	Reverse On	CNTRL	9	
[RVSOFF]	Reverse Off	CNTRL	0	
[UPARROW]	Up Arrow		↑	
[BACKARROW]	Back Arrow		←	
[PI]	PI		π	
[EP]	English Pound		£	

When You See	It Means	You Type		You Will See
[BLACK]	Black	CNTRL	1	
[WHITE]	White	CNTRL	2	
[RED]	Red	CNTRL	3	
[CYAN]	Cyan	CNTRL	4	
[PURPLE]	Purple	CNTRL	5	
[GREEN]	Green	CNTRL	6	
[BLUE]	Blue	CNTRL	7	
[YELLOW]	Yellow	CNTRL	8	
[F1]	Function 1		F1	
[F2]	Function 2	SHIFT	F1	
[F3]	Function 3		F3	
[F4]	Function 4	SHIFT	F3	
[F5]	Function 5		F5	
[F6]	Function 6	SHIFT	F5	
[F7]	Function 7		F7	
[F8]	Function 8	SHIFT	F7	

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** This and the preceding page explain these codes and provide other essential information on entering **Ahoy!** programs. Read these pages **before** entering programs.

BUG REPELLENT

This program will let you debug any *Ahoy!* program. Follow instructions for VIC 20 (cassette or disk) or C-64.

VIC 20 VERSION

By Michael Kleinert and David Barron

For cassette: type in and save the *Bug Repellent* program, then type RUN 63000[RETURN]SYS 828[RETURN]. If you typed the program properly, it will generate a set of two-letter line codes that will match those listed to the right of the respective program lines.

Once you've got a working *Bug Repellent*, type in the program you wish to check. Save it and type the RUN and SYS commands listed above once again, then compare the line codes generated to those listed in the magazine. If you spot a discrepancy, a typing error exists in that line. Important: you must use exactly the same spacing as the program in the magazine. Due to memory limitations on the VIC, the VIC *Bug Repellent* will register an error if your spacing varies from what's printed.

You may type SYS 828 as many times as you wish, but if you use the cassette for anything, type RUN 63000 to restore the *Repellent*.

When your program has been disinfected you may delete all lines from 63000 on. (Be sure the program you type doesn't include lines above 63000!)

For disk: enter *Bug Repellent*, save it, and type RUN:NEW [RETURN]. Type in the program you wish to check, then SYS 828.

To pause the line codes listing, press SHIFT.

To send the list to the printer type OPEN 4,4:CMD 4:SYS 828[RETURN]. When the cursor comes back, type PRINT#4:CLOSE 4[RETURN].

```
•63000 FORX=828TO1023:READY:POKEX,Y:NEXT:END   AC
•63001 DATA169,0,133,63,133,64,165,43,133,251  JL
•63002 DATA165,44,133,252,160,0,132,254,32,228 DF
•63003 DATA3,234,177,251,208,3,76,208,3,230    OE
•63004 DATA251,208,2,230,252,169,244,160,3,32  OH
•63005 DATA30,203,160,0,177,251,170,230,251,20
 8                                             KO
•63006 DATA2,230,252,177,251,32,205,221,169,58 JJ
•63007 DATA32,210,255,169,0,133,253,230,254,32 OK
•63008 DATA228,3,234,165,253,160,0,170,177,251 LG
•63009 DATA201,32,240,6,138,113,251,69,254,170 BP
•63010 DATA138,133,253,177,251,208,226,165,253
 ,41                                           DD
•63011 DATA240,74,74,74,74,24,105,65,32,210    EK
•63012 DATA255,165,253,41,15,24,105,65,32,210  FO
•63013 DATA255,169,13,32,210,255,173,141,2,41  PK
•63014 DATA1,208,249,230,63,208,2,230,64,230   CB
•63015 DATA251,208,2,230,252,76,74,3,169,236   KH
•63016 DATA160,3,32,30,203,166,63,165,64,32    DP
•63017 DATA205,221,169,13,32,210,255,96,230,25
 1                                             EL
•63018 DATA208,2,230,252,96,0,76,73,78,69      OI
•63019 DATA83,58,32,0,76,73,78,69,32,35        FG
•63020 DATA32,0,0,0,0,0                        LE
```

C-64 VERSION

By Michael Kleinert and David Barron

Type in, SAVE, and RUN the *Bug Repellent*. Type NEW, then type in or LOAD the *Ahoy!* program you wish to check. When that's done, SAVE your program (don't RUN it!) and type SYS 49152 [RETURN].

To pause the listing depress and hold the SHIFT key.

Compare the codes your machine generates to the codes listed to the right of the respective program lines. If you spot a difference, an error exists in that line. Jot down the number of lines where contradictions occur, LIST each line, spot the errors, and correct them.

```
•5000 FORX=49152TO49488:READY:POKEX,Y:NEXT:END GJ
•5001 DATA32,161,192,165,43,133,251,165,44,133 DL
•5002 DATA252,160,0,132,254,32,37,193,234,177  DB
•5003 DATA251,208,3,76,138,192,230,251,208,2   OF
•5004 DATA230,252,76,43,192,76,73,78,69,32     KN
•5005 DATA35,32,0,169,35,160,192,32,30,171     CA
•5006 DATA160,0,177,251,170,230,251,208,2,230  CE
•5007 DATA252,177,251,32,205,189,169,58,32,210 JE
•5008 DATA255,169,0,133,253,230,254,32,37,193  CL
•5009 DATA234,165,253,160,0,76,13,193,133,253  NB
•5010 DATA177,251,208,237,165,253,41,240,74,74 MB
•5011 DATA74,74,24,105,65,32,210,255,165,253   EP
•5012 DATA41,15,24,105,65,32,210,255,169,13    GH
•5013 DATA32,220,192,230,63,208,2,230,64,230   AN
•5014 DATA251,208,2,230,252,76,11,192,169,153  NG
•5015 DATA160,192,32,30,171,166,63,165,64,76   BF
•5016 DATA231,192,96,76,73,78,69,83,58,32      EP
•5017 DATA0,169,247,160,192,32,30,171,169,3    PJ
•5018 DATA133,254,32,228,255,201,83,240,6,201  FK
•5019 DATA80,208,245,230,254,32,210,255,169,4  FL
•5020 DATA166,254,160,255,32,186,255,169,0,133 CL
•5021 DATA63,133,64,133,2,32,189,255,32,192    GC
•5022 DATA255,166,254,32,201,255,76,73,193,96  NN
•5023 DATA32,210,255,173,141,2,41,1,208,249    NH
•5024 DATA96,32,205,189,169,13,32,210,255,32   IM
•5025 DATA204,255,169,4,76,195,255,147,83,67   KC
•5026 DATA82,69,69,78,32,79,82,32,80,82        DC
•5027 DATA73,78,84,69,82,32,63,32,0,76         ML
•5028 DATA44,193,234,177,251,201,32,240,6,138  GN
•5029 DATA113,251,69,254,170,138,76,88,192,0   JK
•5030 DATA0,0,0,230,251,208,2,230,252,96       NA
•5031 DATA170,177,251,201,34,208,6,165,2,73    DM
•5032 DATA255,133,2,165,2,208,218,177,251,201  JA
•5033 DATA32,208,212,198,254,76,29,193,0,169   FM
•5034 DATA13,76,210,255,0,0,0                  PA
```

FLANKSPEED FOR THE C-64

By Gordon F. Wheat

Flankspeed will allow you to enter machine language *Ahoy!* programs without any mistakes. Once you have typed the program in, save it for future use. While entering an ML program with *Flankspeed* there is no need to enter spaces or hit the carriage return. This is all done automatically. If you make an error in a line a bell will ring and you will be asked to enter it again. To LOAD in a program Saved with *Flankspeed* use LOAD "name",1,1 for tape, or LOAD "name",8,1 for disk. The function keys may be used after the starting and ending addresses have been entered.

f1—SAVEs what you have entered so far.

f3—LOADs in a program worked on previously.

f5—To continue on a line you stopped on after LOADing in the previously saved work.

f7—Scans through the program to locate a particular line, or to find out where you stopped the last time you entered the program. f7 temporarily freezes the output as well.

```
•5 POKE53280,12:POKE53281,11                   LL
•6 PRINT"[CLEAR][c 8][RVSON][15" "]FLANKSPEED[
 15" "]";                                      ED
•10 PRINT"[RVSON][5" "]MISTAKEPROOF ML ENTRY P
 ROGRAM[6" "]"                                 MC
•15 PRINT"[RVSON][9" "]CREATED BY G. F. WHEAT[
 9" "]"                                        DM
•20 PRINT"[RVSON][3" "]COPR. 1984, ION INTERNA
```

```
 TIONAL INC.[3" "]"                              DH
·30 FORA=54272TO54296:POKEA,0:NEXT               IM
·40 POKE54272,4:POKE54273,48:POKE54277,0:POKE5
 4278,249:POKE54296,15                           NH
·70 FORA=680TO699:READB:POKEA,B:NEXT             KO
·75 DATA169,251,166,253,164,254,32,216,255,96    HJ
·76 DATA169,0,166,251,164,252,32,213,255,96      JB
·80 B$="STARTING ADDRESS IN HEX":GOSUB2010:AD=
 B:SR=B                                          HC
·85 GOSUB2520:IFB=0THEN80                        FO
·86 POKE251,T(4)+T(3)*16:POKE252,T(2)+T(1)*16    KE
·90 B$="ENDING ADDRESS IN HEX":GOSUB2010:EN=B    IF
·95 GOSUB2510:IFB=0THEN80                        FP
·96 POKE254,T(2)+T(1)*16:B=T(4)+1+T(3)*16        MN
·97 IFB>255THENB=B-255:POKE254,PEEK(254)+1       GE
·98 POKE253,B:PRINT                              HN
·100 REM  GET HEX LINE                           IL
·110 GOSUB3010:PRINT": [c P][LEFT]";:FORA=0TO8   FG
·120 FORB=0TO1:GOTO210                           MD
·125 NEXTB                                       ME
·130 A%(A)=T(1)+T(0)*16:IFAD+A-1=ENTHEN310       LH
·135 PRINT" [c P][LEFT]";                        IK
·140 NEXTA:T=AD-(INT(AD/256)*256):PRINT" "       PD
·150 FORA=0TO7:T=T+A%(A):IFT>255THENT=T-255      LK
·160 NEXT                                        IA
·170 IFA%(8)<>TTHENGOSUB1010:GOTO110             FK
·180 FORA=0TO7:POKEAD+A,A%(A):NEXT:AD=AD+8:GOT
 O110                                            MN
·200 REM     GET HEX INPUT                       AB
·210 GETA$:IFA$=""THEN210                        HO
·211 IFA$=CHR$(20)THEN270                        GC
·212 IFA$=CHR$(133)THEN4000                      MD
·213 IFA$=CHR$(134)THEN4100                      KF
·214 IFA$=CHR$(135)THENPRINT" ":GOTO4500         GE
·215 IFA$=CHR$(136)THENPRINT" ":GOTO4700         BJ
·220 IFA$>"@"ANDA$<"G"THENT(B)=ASC(A$)-55:GOTO
 250                                             GM
·230 IFA$>"/"ANDA$<":"THENT(B)=ASC(A$)-48:GOTO
 250                                             LE
·240 GOSUB1100:GOTO210                           LL
·250 PRINTA$"[c P][LEFT]";                       OA
·260 GOTO125                                     CG
·270 IFA>0THEN280                                OP
·272 A=-1:IFB=1THEN290                           OB
·274 GOTO140                                     CJ
·280 IFB=0THENPRINTCHR$(20);CHR$(20);:A=A-1      HG
·285 A=A-1                                       BE
·290 PRINTCHR$(20);:GOTO140                      KH
·300 REM LAST LINE                               AD
·310 PRINT" ":T=AD-(INT(AD/256)*256)             GJ
·320 FORB=0TOA-1:T=T+A%(B):IFT>255THENT=T-255    PL
·330 NEXT                                        IA
·340 IFA%(A)<>TTHENGOSUB1010:GOTO110             KF
·350 FORB=0TOA-1:POKEAD+B,A%(B):NEXT             HN
·360 PRINT:PRINT"YOU ARE FINISHED!":GOTO4000     ON
·1000 REM BELL AND ERROR MESSAGES                FL
·1010 PRINT:PRINT"LINE ENTERED INCORRECTLY":PR
 INT:GOTO1100                                    DH
·1020 PRINT:PRINT"INPUT A 4 DIGIT HEX VALUE!":
 GOTO1100                                        JA
·1030 PRINT:PRINT"ENDING IS LESS THAN STARTING
 !":B=0:GOTO1100                                 HD
·1040 PRINT:PRINT"ADDRESS NOT WITHIN SPECIFIED
 RANGE!":B=0:GOTO1100                            AG
·1050 PRINT:PRINT"NOT ZERO PAGE OR ROM!":B=0:G
 OTO1100                                         KN
·1060 PRINT"?ERROR IN SAVE":GOTO1100             EI
·1070 PRINT"?ERROR IN LOAD":GOTO1100             GL
·1080 PRINT:PRINT:PRINT"END OF ML AREA":PRINT    PG
·1100 POKE54276,17:POKE54276,16:RETURN           BH
·1200 OPEN15,8,15:INPUT#15,A,A$:CLOSE15:PRINTA
 $:RETURN                                        IM
·2000 REM GET FOUR DIGIT HEX                     PC
·2010 PRINT:PRINTB$;:INPUTT$                     GM
·2020 IFLEN(T$)<>4THENGOSUB1020:GOTO2010         II
·2040 FORA=1TO4:A$=MID$(T$,A,1):GOSUB2060:IFT(
 A)=16THENGOSUB1020:GOTO2010                     AD
·2050 NEXT:B=(T(1)*4096)+(T(2)*256)+(T(3)*16)+
 T(4):RETURN                                     GF
·2060 IFA$>"@"ANDA$<"G"THENT(A)=ASC(A$)-55:RET
 URN                                             EH
·2070 IFA$>"/"ANDA$<":"THENT(A)=ASC(A$)-48:RET
 URN                                             KP
·2080 T(A)=16:RETURN                             NP
·2500 REM ADRESS CHECK                           LI
·2510 IFAD>ENTHEN1030                            MI
·2515 IFB<SRORB>ENTHEN1040                       MG
·2520 IFB<256OR(B>40960ANDB<49152)ORB>53247THE
 N1050                                           MI
·2530 RETURN                                     IM
·3000 REM ADDRESS TO HEX                         EB
·3010 AC=AD:A=4096:GOSUB3070                     HG
·3020 A=256:GOSUB3070                            CE
·3030 A=16:GOSUB3070                             PN
·3040 A=1:GOSUB3070                              MJ
·3060 RETURN                                     IM
·3070 T=INT(AC/A):IFT>9THENA$=CHR$(T+55):GOTO3
 090                                             CJ
·3080 A$=CHR$(T+48)                              JP
·3090 PRINTA$;:AC=AC-A*T:RETURN                  AC
·4000 A$="**SAVE**":GOSUB4200                    AI
·4050 OPEN1,T,1,A$:SYS680:CLOSE1                 LH
·4060 IFST=0THENEND                              EO
·4070 GOSUB1060:IFT=8THENGOSUB1200               FJ
·4080 GOTO4000                                   FF
·4100 A$="**LOAD**":GOSUB4200                    AB
·4150 OPEN1,T,0,A$:SYS690:CLOSE1                 MF
·4160 IFST=64THEN110                             JH
·4170 GOSUB1070:IFT=8THENGOSUB1200               CM
·4180 GOTO4100                                   FO
·4200 PRINT" ":PRINTTAB(14)A$                    FG
·4210 PRINT:A$="":INPUT"FILENAME";A$             OM
·4215 IFA$=""THEN4210                            GF
·4220 PRINT:PRINT"TAPE OR DISK?":PRINT           DF
·4230 GETB$:T=1:IFB$="D"THENT=8:A$="@0:"+A$:RE
 TURN                                            IG
·4240 IFB$<>"T"THEN4230                          FN
·4250 RETURN                                     IM
·4500 B$="CONTINUE FROM ADDRESS":GOSUB2010:AD=
 B                                               DK
·4510 GOSUB2515:IFB=0THEN4500                    MA
·4520 PRINT:GOTO110                              OI
·4700 B$="BEGIN SCAN AT ADDRESS":GOSUB2010:AD=
 B                                               FH
·4705 GOSUB2515:IFB=0THEN4700                    NK
·4706 PRINT:GOTO4740                             DI
·4710 FORB=0TO7:AC=PEEK(AD+B):GOSUB3030:IFAD+B
 =ENTHENAD=SR:GOSUB1080:GOTO110                  BK
·4715 PRINT" ";:NEXTB                            EC
·4720 PRINT:AD=AD+8                              GN
·4730 GETB$:IFB$=CHR$(136)THEN110                MN
·4740 GOSUB3010:PRINT": ";:GOTO4710              JD
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering ***Ahoy!*** programs. Refer to these pages **before** entering any programs!

MOVING THROUGH *Color Memory*

FROM PAGE 18

MENU FOR VIC 20

```
·1 REM MENU FOR VIC--POKEING COLOR MEMORY JG
·7 REM JD
·8 REM DISABLE RUN/STOP & SHIFT/COMMODORE EG
·9 REM JD
·10 POKE 808,100:POKE 657,128 JP
·20 DIM CO(4) KH
·27 REM JD
·28 REM SET CO() TO ADDRESSES OF OPTION F
 IELDS IN COLOR MEMORY MD
·29 REM JD
·30 CM=38400:CO(0)=38513:FOR I=1 TO 4:CO(
 I)=CO(I-1)+22:NEXT PF
·37 REM JD
·38 REM SET OLD AND NEW COLOR LOCATIONS A
 ND COLOR VALUES IL
·40 NP=2:XP=2:NC=1:XC=0:POKE 36879,232 PC
·41 PRINT "[BLACK]"; KF
·47 REM JD
·48 REM PRINT THE MENU AO
·49 REM JD
·50 CR$=CHR$(13):PRINT "[CLEAR] CHOOSE YO
 UR SANDWICH" EN
·51 PRINT "[4" "]F5=UP  F7=DOWN"CR$"[5" "
 ]SPACE=CHOICE" BE
·52 PRINT "[21"*"]"CR$"*[19" "]*" FF
·53 PRINT "*  ROAST BEEF[7" "]*"CR$"*  CO
 RNED BEEF[6" "]*" LB
·54 PRINT "*  SHAVED HAM[7" "]*"CR$"*  PA
 STRAMI[9" "]*" HN
·55 PRINT "*  TURKEY & SALAMI  *"CR$"*[19
 " "]*" KG
·56 PRINT "[21"*"]"CR$CR$CR$CR$CR$ GB
·57 PRINT "[4" "]PRESS [RVSON]Q[RVSOFF] T
 O QUIT"; BJ
·60 GOSUB 900 DD
·97 REM JD
·98 REM READ KEYBOARD BA
·99 REM JD
·100 A=PEEK(197):IF A=64 THEN 100 CH
·106 REM JD
·107 REM JD
·108 REM STOP NOW? BI
·109 REM JD
·110 IF A=48 THEN 800 DJ
·117 REM JD
·118 REM CHOICE MADE? JB
·119 REM JD
·120 IF A=32 THEN GOSUB 200:GOTO 100 NH
·127 REM JD
·128 REM MOVE UP? KN
·129 REM JD
·130 IF A=55 THEN NP=NP-1:IF NP<0 THEN NP
 =4 BB
·137 REM JD
·138 REM MOVE DOWN? EN
·139 REM JD
·140 IF A=63 THEN NP=NP+1:IF NP>4 THEN NP
 =0 AD
·148 REM IF MOVE, GO DO IT GC
·150 IF NP<>XP THEN GOSUB 920:GOSUB 900:X
 P=NP AO
·160 POKE 197,64:GOTO 100 CN
·200 IF PEEK(197)<>64 THEN 200 DL
·207 REM JD
·208 REM CLEAR OLD CHOICE JJ
·209 REM JD
·210 PRINT "[HOME][13"[DOWN]"]YOU CHOSE:"
 CR$"[21" "][UP]" FE
·217 REM JD
·218 REM JUMP TABLE CL
·219 REM JD
·220 ON NP+1 GOTO 300,400,500,600,700 KM
·300 PRINT "ROAST BEEF ON WHEAT":RETURN EM
·400 PRINT "CORNED BEEF ON RYE":RETURN OJ
·500 PRINT "SHAVED HAM ON BUN":RETURN BK
·600 PRINT "PASTRAMI ON TOAST":RETURN EP
·700 PRINT "TURKEY/SALAMI ON ROLL":RETURN HI
·797 REM JD
·798 REM QUIT ROUTINE PD
·799 REM JD
·800 IF PEEK(197)<>64 THEN 800 EB
·810 PRINT "[CLEAR]":POKE 808,112:POKE 65
 7,0:POKE 198,0:END GL
·897 REM JD
·898 REM POKE COLORS INTO COLOR MEMORY PD
·899 REM JD
·900 FOR I=CO(NP) TO CO(NP)+15:POKE I,NC:
 NEXT:RETURN LK
·920 FOR I=CO(XP) TO CO(XP)+15:POKE I,XC:
 NEXT:RETURN II
```

MENU FOR C-64

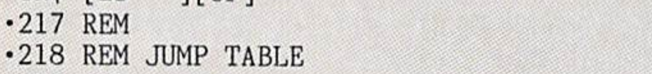
```
·1 REM MENU FOR 64--POKEING COLOR MEMORY PO
·7 REM JD
·8 REM DISABLE RUN/STOP AND RESTORE JC
·9 REM JD
·10 POKE 792,193:POKE 808,239:POKE 657,12
 8 MF
·20 DIM CO(4) JL
·27 REM JD
·28 REM SET CO() TO ADDRESSES OF OPTION F
 IELDS IN COLOR MEMORY DP
·29 REM JD
```

```
•30 CM=55296:CO(0)=55499:FOR I=1 TO 4:CO(
I)=CO(I-1)+40:NEXT                          FJ
•37 REM                                     JD
•38 REM SET OLD AND NEW COLOR LOCATIONS A
ND COLOR VALUES                             FE
•40 NP=2:XP=2:NC=1:XC=14                    NL
•47 REM                                     JD
•48 REM PRINT THE MENU                      EE
•49 REM                                     JD
•50 CR$=CHR$(13):PRINT "[CLEAR][10" "]CHO
OSE YOUR SANDWICH"                          HD
•51 PRINT "[5" "]F5=UP[3" "]F7=DOWN[3" "]
SPACE=CHOICE"                               AF
•52 PRINT CR$"[23"*"]"CR$"*[21" "]*"        KM
•53 PRINT "*  ROAST BEEF[9" "]*"CR$"*  CO
RNED BEEF[8" "]*"                           OF
•54 PRINT "*  SHAVED HAM[9" "]*"CR$"*  PA
STRAMI[11" "]*"                             ON
•55 PRINT "*  TURKEY & SALAMI[4" "]*"CR$"
*[21" "]*"                                  MO
•56 PRINT "[23"*"]"CR$CR$CR$CR$CR$CR$       EA
•57 PRINT "[9" "]PRESS [RVSON]RUN/STOP[RV
SOFF] TO QUIT";                             LI
•60 GOSUB 900                               DB
•97 REM                                     JD
•98 REM READ KEYBOARD                       AI
•99 REM                                     JD
•100 A=PEEK(197):IF A=64 THEN 100           PP
•106 REM                                    JD
•107 REM                                    JD
•108 REM STOP NOW?                          BK
•109 REM                                    JD
•110 IF A=63 THEN 800                       DM
•117 REM                                    JD
•118 REM CHOICE MADE?                       LJ
•119 REM                                    JD
•120 IF A=60 THEN GOSUB 200:GOTO 100        OH
•127 REM                                    JD
•128 REM MOVE UP?                           KJ
•129 REM                                    JD
•130 IF A=6 THEN NP=NP-1:IF NP<0 THEN NP=
4                                           CJ
•137 REM                                    JD
•138 REM MOVE DOWN?                         DN
•139 REM                                    JD
•140 IF A=3 THEN NP=NP+1:IF NP>4 THEN NP=
0                                           FF
•148 REM IF MOVE, GO DO IT                  DJ
•150 IF NP<>XP THEN GOSUB 920:GOSUB 900:X
P=NP                                        AL
•160 POKE 197,64:GOTO 100                   DN
•200 IF PEEK(197)<>64 THEN 200              FF
•207 REM                                    JD
•208 REM CLEAR OLD CHOICE                   GF
•209 REM                                    JD
•210 PRINT "[HOME][13"[DOWN]"]YOU CHOSE:"
CR$"[25" "][UP]"                            CG
•217 REM                                    JD
•218 REM JUMP TABLE                         EB
•219 REM                                    JD
•220 ON NP+1 GOTO 300,400,500,600,700       GC
•300 PRINT "ROAST BEEF ON WHEAT":RETURN     PE
•400 PRINT "CORNED BEEF ON RYE":RETURN      EL
•500 PRINT "SHAVED HAM ON A CROISSANT":RE
TURN                                        KG
•600 PRINT "PASTRAMI ON TOAST":RETURN       DI
•700 PRINT "TURKEY & SALAMI ON ROLL":RETU
RN                                          LH
•797 REM                                    JD
•798 REM QUIT ROUTINE                       PJ
•799 REM                                    JD
•800 IF PEEK(197)<>64 THEN 800              EL
•810 PRINT "[CLEAR]":POKE 792,71:POKE 808
,237:POKE 657,0:POKE 198,0:END              OE
•897 REM                                    JD
•898 REM POKE COLORS INTO COLOR MEMORY      BM
•899 REM                                    JD
•900 FOR I=CO(NP) TO CO(NP)+15:POKE I,NC:
NEXT:RETURN                                 JD
•920 FOR I=CO(XP) TO CO(XP)+15:POKE I,XC:
NEXT:RETURN                                 LJ
```

FAST MENU FOR C-64

```
•7 REM                                      JD
•8 REM DISABLE RUN/STOP AND RESTORE         JC
•9 REM                                      JD
•10 POKE 792,193:POKE 808,239:POKE 657,12
8                                           MF
•20 DIM V$(23)                              LF
•27 REM                                     JD
•28 REM SET V$() TO ROW NUMBERS             BG
•29 REM                                     JD
•30 V$(0)="[HOME]":FOR I=1 TO 23:V$(I)=V$
(I-1)+"[DOWN]":NEXT                         MB
•37 REM                                     JD
•38 REM SET OLD AND NEW COLOR LOCATIONS A
ND COLOR STRINGS                            IK
•40 NP=2:XP=1:XC$="[c 7][39","]"            MJ
•41 HC$="[WHITE]"+RIGHT$(XC$,39)            JG
•47 REM                                     JD
•48 REM PRINT THE MENU                      EE
•49 REM                                     JD
•50 CR$=CHR$(13):PRINT "[CLEAR][10" "]CHO
OSE YOUR SANDWICH"                          HD
•51 PRINT "[5" "]F5=UP[3" "]F7=DOWN[3" "]
SPACE=CHOICE"                               AF
•52 PRINT CR$"[23"*"]"CR$"*[21" "]*"        KM
•53 PRINT "*  ROAST BEEF[9" "]*"CR$"*  CO
RNED BEEF[8" "]*"                           OF
•54 PRINT "*  SHAVED HAM[9" "]*"CR$"*  PA
STRAMI[11" "]*"                             ON
•55 PRINT "*  TURKEY & SALAMI[4" "]*"CR$"
*[21" "]*"                                  MO
•56 PRINT "[23"*"]"CR$CR$CR$CR$CR$CR$       EA
•57 PRINT "[9" "]PRESS [RVSON]RUN/STOP[RV
SOFF] TO QUIT";                             LI
```

```
·60 GOSUB 900:XP=NP                        GC
·97 REM                                    JD
·98 REM READ KEYBOARD                      AI
·99 REM                                    JD
·100 A=PEEK(197):IF A=64 THEN 100          PP
·107 REM                                   JD
·108 REM QUIT?                             AL
·109 REM                                   JD
·110 IF A=63 THEN 800                      DM
·117 REM                                   JD
·118 REM CHOICE MADE?                      LJ
·119 REM                                   JD
·120 IF A=60 THEN GOSUB 200:GOTO 100       OH
·127 REM                                   JD
·128 REM MOVE UP?                          KJ
·129 REM                                   JD
·130 IF A=6 THEN NP=NP-1:IF NP<0 THEN NP=
4                                          CJ
·137 REM                                   JD
·138 REM MOVE DOWN?                        DN
·139 REM                                   JD
·140 IF A=3 THEN NP=NP+1:IF NP>4 THEN NP=
0                                          FF
·147 REM                                   JD
·148 REM IF MOVE, GO DO IT                 DJ
·149 REM                                   JD
·150 IF NP<>XP THEN GOSUB 900:XP=NP        GG
·160 POKE 197,64:GOTO 100                  DN
·200 IF PEEK(197)<>64 THEN 200             FF
·207 REM                                   JD
·208 REM CLEAR OLD CHOICE                  GF
·209 REM                                   JD
·210 PRINT "[HOME][13"[DOWN]"]YOU CHOSE:"
CR$"[25" "][UP]"                           CG
·217 REM                                   JD
·218 REM JUMP TABLE                        EB
·219 REM                                   JD
·220 ON NP+1 GOTO 300,400,500,600,700      GC
·300 PRINT "ROAST BEEF ON WHEAT":RETURN    PE
·400 PRINT "CORNED BEEF ON RYE":RETURN     EL
·500 PRINT "SHAVED HAM ON A CROISSANT":RE
TURN                                       KG
·600 PRINT "PASTRAMI ON TOAST":RETURN      DI
·700 PRINT "TURKEY & SALAMI ON ROLL":RETU
RN                                         LH
·797 REM                                   JD
·798 REM RESTORE AND END                   HJ
·799 REM                                   JD
·800 IF PEEK(197)<>64 THEN 800             EL
·810 PRINT "[CLEAR]":POKE 792,71:POKE 808
,237:POKE 657,0:POKE 198,0:END             OE
·897 REM                                   JD
·898 REM PRINT COLORS USING FALSE SCREEN
MEMORY                                     CE
·900 POKE 648,60:PRINT V$(NP+5)HC$V$(XP+5
)XC$:POKE 648,4:RETURN                     FK
```



WORM FOR VIC 20

```
·1 REM WORM FOR VIC--ANIMATE WITH COLOR M
EMORY                                      OE
·10 POKE 808,100:POKE 657,128              JP
·20 DIM WM(14)                             OC
·30 CM=38400:WM(0)=38520:WM(1)=38521:WM(2
)=38544:LW=2                               KB
·40 CW=2:CB=0:CF=5:POKE 36879,8:PRINT CHR
$(144)CHR$(147);                           HF
·50 F$="[21"[s Q]"]"                       BF
·51 FOR I=0 TO 23:PRINT F$:NEXT:PRINT F$"
[HOME]";                                   BG
·52 FOR I=CM+21 TO CM+505 STEP 22:POKE I,
0:NEXT                                     LD
·53 FOR I=7701 TO 8185 STEP 22:POKE I,81:
NEXT                                       GC
·60 FOR I=0 TO LW:POKE WM(I),CW:NEXT       LM
·70 FOR I=0 TO 11                          ML
·75 F=CM+INT(506*RND(9)):IF (PEEK(F)AND 1
5)<>CB THEN 75                             CD
·80 POKE F,CF:NEXT                         FM
·100 A=PEEK(197):B=PEEK(653):IF A=24 OR A
=48 THEN 800                               ID
·110 IF A<>63 AND A<>55 AND B<>1 AND B<>2
 THEN 100                                  EC
·120 XW=WM(LW):FOR I=LW TO 1 STEP -1:WM(I
)=WM(I-1):NEXT                             FL
·130 IF A=55 THEN WM(0)=WM(0)-22:IF WM(0)
<CM THEN WM(0)=WM(0)+506                   EI
·135 IF A=63 THEN WM(0)=WM(0)+22:IF WM(0)
>38905 THEN WM(0)=WM(0)-506                MG
·140 IF B=2 THEN WM(0)=WM(0)-1:IF WM(0)<C
M THEN WM(0)=WM(0)+506                     OP
·145 IF B=1 THEN WM(0)=WM(0)+1:IF WM(0)>3
8905 THEN WM(0)=WM(0)-506                  HJ
·147 REM                                   JD
·148 REM IS IT FOOD?                       EN
·149 REM                                   JD
·150 IF (PEEK(WM(0))AND 15)=CF THEN LW=LW
+1:WM(LW)=XW:GOTO 170                      OE
·160 POKE XW,CB                            AH
·170 POKE WM(0),CW:IF LW>13 THEN 30        FJ
·180 GOTO 100                              BP
·800 IF PEEK(197)<>64 THEN 800             EB
·810 PRINT "[c 7][CLEAR]":POKE 808,112:PO
KE 657,0:POKE 198,0:SYS 65234:END          BM
```

WORM FOR C-64

```
·1 REM WORM FOR 64--ANIMATE WITH COLOR ME
MORY                                       FO
·7 REM                                     JD
·8 REM DISABLE RUN/STOP AND RESTORE        JC
·9 REM                                     JD
·10 POKE 792,193:POKE 808,239:POKE 657,12
8                                          MF
·20 DIM WM(14)                             OB
·27 REM                                    JD
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
•28 REM SET WM() TO WORM'S STARTING POSIT
 ION IN COLOR MEMORY                       PG
•29 REM                                    JD
•30 CM=55296:WM(0)=55712:WM(1)=55713:WM(2
 )=55752:LW=2                              NF
•37 REM                                    JD
•38 REM SET WORM, FOOD, AND BACKGROUND CO
 LOR VALUES                                JO
•40 CW=7:CB=9:CF=5:POKE 53281,CB:PRINT CH
 R$(149)CHR$(147);                         KP
•47 REM                                    JD
•48 REM FILL THE SCREEN WITH DOTS          BB
•49 REM                                    JD
•50 F$="[39"[s Q]"]"                       JC
•51 FOR I=0 TO 23:PRINT F$:NEXT:PRINT F$"
 [HOME]";                                  JM
•52 FOR I=1063 TO 2023 STEP 40:POKE I,81:
 NEXT                                      IK
•57 REM                                    JD
•58 REM PUT WORM ON SCREEN                 GN
•59 REM                                    JD
•60 FOR I=0 TO LW:POKE WM(I),CW:NEXT       HN
•67 REM                                    JD
•68 REM PUT FOOD OUT RANDOMLY              BP
•69 REM                                    JD
•70 FOR I=0 TO 11                          MH
•75 F=CM+INT(1000*RND(9)):IF (PEEK(F)AND
 15)<>CB THEN 75                           NJ
•80 POKE F,CF:NEXT                         GN
•97 REM                                    JD
•98 REM READ KEYBOARD                      AI
•99 REM                                    JD
•100 A=PEEK(197):B=PEEK(653):IF A=63 THEN
  800                                      BD
•106 REM                                   JD
•107 REM                                   JD
•108 REM LEGAL MOVEMENT?                   LD
•109 REM                                   JD
•110 IF A<>6 AND A<>3 AND B<>1 AND B<>2 T
 HEN 100                                   NF
•117 REM                                   JD
•118 REM SHIFT ADDRESSES                   MA
•119 REM                                   JD
•120 XW=WM(LW):FOR I=LW TO 1 STEP -1:WM(I
 )=WM(I-1):NEXT                            KN
•127 REM                                   JD
•128 REM MOVE UP?                          KJ
•129 REM                                   JD
•130 IF A=6 THEN WM(0)=WM(0)-40:IF WM(0)<
 CM THEN WM(0)=WM(0)+1000                  AH
•132 REM                                   JD
•133 REM MOVE DOWN?                        DN
•134 REM                                   JD
•135 IF A=3 THEN WM(0)=WM(0)+40:IF WM(0)>
 56295 THEN WM(0)=WM(0)-1000               JF
•137 REM                                   JD
•138 REM MOVE LEFT?                        BE
•139 REM                                   JD
•140 IF B=2 THEN WM(0)=WM(0)-1:IF WM(0)<C
 M THEN WM(0)=WM(0)+1000                   JJ
•142 REM                                   JD
•143 REM MOVE RIGHT?                       IC
•144 REM                                   JD
•145 IF B=1 THEN WM(0)=WM(0)+1:IF WM(0)>5
 6295 THEN WM(0)=WM(0)-1000                CG
•147 REM                                   JD
•148 REM IS IT FOOD?                       DP
•149 REM                                   JD
•150 IF (PEEK(WM(0))AND 15)=CF THEN LW=LW
 +1:WM(LW)=XW:GOTO 170                     JH
•158 REM ERASE WORM'S TAIL                 PH
•159 REM                                   JD
•160 POKE XW,CB                            AJ
•167 REM                                   JD
•168 REM COLOR WORM'S HEAD                 OA
•169 REM                                   JD
•170 POKE WM(0),CW:IF LW>13 THEN 30        EC
•180 GOTO 100                              CF
•797 REM                                   JD
•798 REM QUIT ROUTINE                      PJ
•799 REM                                   JD
•800 IF PEEK(197)<>64 THEN 800             EL
•810 POKE 53281,6:PRINT "[c 7][CLEAR]":PO
 KE 792,71:POKE 808,237:POKE 657,0:POKE 1
 98,0:END                                  NL
```

ASSEMBLER ESCAPADES

FROM PAGE 55 LISTING 1

```
•1 IF QQ=1 THEN 140                        FO
•2 IF QQ=2 THEN END                        DJ
•4 :REM                                    ML
•5 :REM -- SAVE SCREEN TO DISK --          KE
•6 :REM =RUPERT-REPORT=LISTING-1=          JA
•7 :REM                                    ML
•10 DEF FNH(X)=INT(X/256) :REM MSB OF X    HF
•20 DEF FNL(X)=X-256*FNH(X) :REM LSB       DF
•25 :REM                                   ML
•30 GOSUB 500  :REM: PUT IMAGE ON SCREEN   PN
•35 :REM                                   ML
•40 FLNAM$="@:SCRN1"                       IA
•45 FA$=FLNAM$                             ON
•50 SADDR=1024 : EADDR=2024                CL
•60 GOSUB 220  :REM: SAVE SCREEN MEMORY    FL
•65 :REM                                   ML
•70 FLNAM$="@:COLR1"                       EC
•75 FB$=FLNAM$                             PM
•80 SADDR=55296 : EADDR=56296              JK
•90 GOSUB 220  :REM: SAVE COLOR MEMORY     NP
•100 :REM -- RESTORE SCREEN --             DA
```

```
·110 PRINT CHR$(147):PRINT"PRESS ANY KEY
 TO RESTORE SCREEN"                           GM
·120 GET AA$ : IF AA$="" THEN 120             NF
·130 QQ=1 : FLNAM$=FB$ : GOTO 600             CJ
·140 QQ=2 : FLNAM$=FA$ : GOTO 600             CL
·150 REM =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= CD
·160 REM   USE KERNAL 'SAVE' ROUTINE          LD
·170 REM   TO SAVE MEMORY FROM (SADDR)        DG
·180 REM   TO (EADDR-1) IN FILE "FLNAM$"      BO
·190 REM =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= CD
·200 REM                                      JD
·210 REM                                      JD
·220 AREG=780 : XREG=781 : YREG=782           HE
·230 OPEN 8,8,8,FLNAM$                        GP
·240 SVE=65496    :REM $FFD8                  ME
·250 ZPTR=251     :REM $FB (ZERO PAGE)        JG
·260 POKE ZPTR+1,FNH(SADDR)                   MK
·270 POKE ZPTR,FNL(SADDR)                     NO
·280 POKE AREG,ZPTR                           DE
·290 POKE XREG,FNL(EADDR)                     KK
·300 POKE YREG,FNH(EADDR)                     KN
·310 SYS SVE                                  IL
·320 CLOSE 8                                  NI
·330 RETURN                                   IM
·494 :REM                                     ML
·495 :REM << FILL SCREEN >>                   ML
·500 FOR MEM=1024 TO 2022 STEP 3              CC
·510 PRINT CHR$(168)CHR$(127)CHR$(168);       PP
·520 POKE MEM+54272,RND(0)*16                 GH
·530 NEXT                                     IA
·540 RETURN                                   IM
·594 :REM -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= FO
·595 :REM  LOAD FLNAM$ BACK INTO MEMORY       KL
·596 :REM  EXECUTION RESUMES AT START         JO
·597 :REM          OF PROGRAM                 IP
·598 :REM -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= FO
·600 LOAD FLNAM$,8,1                          NB
```

LISTING 2

```
·10 REM                                       JD
·20 REM =-=ASSEMBLER SOURCE CODE=-=           JH
·30 REM                                       JD
·210 .;-- SAVE SCREEN TO DISK --              DM
·220 .       ORG $C000                        CH
·230 .       JMP :ENTRY                       PH
·240 .:SAVE  .EQ $FFD8  ;S.A. OF 'SAVE'       GO
·243 .:SADDR                ;START SAVING     CI
·246 .       .DB $04 $00                      PG
·250 .:EADDR                ;END SAVING       EO
·253 .       .DB $07 $E8                      DA
·256 .:ENTRY                                  AB
·260 .       LDA :SADDR      ;HI BYTE         KL
·265 .       STA $FC                          NO
·270 .       LDA :SADDR+$01  ;LO BYTE         PM
·275 .       STA $FB                          NP
·280 .       LDA #$FB        ;ZERO PAGE       EE
·290 .       LDX :EADDR+$01  ;LO BYTE         HD
·300 .       LDY :EADDR      ;HI BYTE         PJ
·310 .       JSR :SAVE                        OJ
·320 .       RTS             ;TO BASIC        HL
·330 .:END                                    OL
·340 .         .EN                            PE
```

LISTING 3

```
·10 REM                                       JD
·20 REM =-= SYMBOL TABLE =-=                  AH
·30 REM                                       JD
·40 $FFD8 :SAVE     $C003 :SADDR              LM
·50 $C005 :EADDR    $C007 :ENTRY              MA
·60 $C01D :END      $C004 :SADDR+$01          NA
·70 $C006 :EADDR+$01                          PD
```

LISTING 4

```
·10 REM                                       JD
·20 REM =-=DISASSEMBLED OBJECT CODE=-=        CL
·30 REM                                       JD
·230 C000 4C 07 C0 JMP $C007                  AD
·246 C003 04          ***                     DA
·247 C004 00          BRK                     CI
·253 C005 07          ***                     EJ
·254 C006 E8          INX                     EL
·260 C007 AD 03 C0 LDA $C003                  BA
·265 C00A 85 FC    STA $FC                    IN
·270 C00C AD 04 C0 LDA $C004                  DE
·275 C00F 85 FB    STA $FB                    IO
·280 C011 A9 FB    LDA #$FB                   GA
·290 C013 AE 06 C0 LDX $C006                  CN
·300 C016 AC 05 C0 LDY $C005                  GH
·310 C019 20 D8 FF JSR $FFD8                  KM
·320 C01C 60       RTS                        FC
```

LISTING 5

```
·5 REM=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= EO
·6 REM  INSERT THESE STATEMENTS INTO          AJ
·7 REM           LISTING 1                    OA
·8 REM=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= EO
·60 GOSUB 190                                 CO
·90 GOSUB 230                                 CJ
·190 MEM=49152 :REM  ORG=$C000 (49152)        HK
·200 READ BYTE$:IF BYTE$="DONE" THEN 230      HF
·210 GOSUB 1000 :REM CONVERT TO DECIMAL       PD
·220 POKE MEM,BYTE : MEM=MEM+1 :GOTO 200      OF
·230 OPEN 8,8,8,FLNAM$                        GP
·240 SS=49155 :REM   STORE SADDR @ $C003      DO
·250 EE=49157 :REM   STORE EADDR @ $C005      DI
·260 POKE SS,FNH(SADDR)                       AN
·270 POKE SS+1,FNL(SADDR)                     AP
·280 POKE EE,FNH(EADDR)                       PL
·290 POKE EE+1,FNL(EADDR)                     NB
·300 :REM   ORG IS $C000 (49152)              JF
·310 SYS 49152                                KF
·320 CLOSE 8                                  NI
·330 RETURN                                   IM
·400 DATA 4C,07,C0,04,00,07,E8                MA
·410 DATA AD,03,C0,85,FC                      IN
·420 DATA AD,04,C0,85,FB,A9,FB                BE
·430 DATA AE,06,C0,AC,05,C0                   PC
·440 DATA 20,D8,FF,60                         NG
·450 DATA DONE                                LH
·1000 REM - CONVERT 2 DIGIT HEX TO DEC -      MA
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering ***Ahoy!*** programs. Refer to these pages **before** entering any programs!

```
·1010 BYTE=0                                  BA
·1020 FOR NN=1 TO 2                           MB
·1030 DD=ASC(MID$(BYTE$,NN,1))-48             BK
·1040 DD=DD+7*(DD>16)                         EL
·1050 BYTE=16*BYTE+DD                         EE
·1060 NEXT                                    IA
·1070 RETURN                                  IM
```

By the Numbers

FROM PAGE 47

```
·1 REM ********************************** DN
·5 REM ******** LISTING NO. 1 ********** GD
·10 REM ***** BY THE NUMBERS *********** IN
·20 REM * A NUMBER CONVERSION PROGRAM ** JG
·22 REM ******************************** DD
·25 DIM HEX$(8),BIT$(8),BIT(8),H$(16),B$(
 16),TEMP$(2)                            EF
·27 REM SET UP HEX-BINARY CONVERSION TABL
 ES:                                     AK
·35 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F BK
·36 DATA 0000,0001,0010,0011,0100,0101,01
 10                                      CD
·37 DATA 0111,1000,1001,1010,1011,1100,11
 01,1110,1111                            AO
·38 FOR L=1 TO 16:READ H$(L):NEXT L      KJ
·39 FOR L=1 TO 16:READ B$(L):NEXT L      JP
·40 REM CLEAR SCREEN AND PRINT MENU:     HB
·45 PRINT CHR$(147):PRINT:PRINT "THIS PRO
 GRAM WILL CONVERT:"                     PE
·50 PRINT:PRINT "[3" "](A) DECIMAL TO HEX
 ADECIMAL"                               JL
·60 PRINT "[3" "](B) HEXADECIMAL TO DECIM
 AL"                                     OO
·70 PRINT:PRINT "[3" "](C) BINARY TO DECI
 MAL"                                    JF
·80 PRINT "[3" "](D) DECIMAL TO BINARY"  GA
·90 PRINT:PRINT "[3" "](E) HEXADECIMAL TO
  BINARY"                                GP
·100 PRINT "[3" "](F) BINARY TO HEXADECIM
 AL":PRINT                               BI
·110 A$="":INPUT "WHAT CONVERSION DO YOU
 WANT (A-F)";A$                          PO
·115 IF A$="" THEN 110                   CH
·116 IF LEN(A$)<>1 THEN 110              AG
·117 IF A$<"A" OR A$>"F" THEN 110        OJ
·120 A=ASC(A$)-64:REM TRANSLATE A$ INTO A
 N INTEGER FROM 1 (A) TO 6 (F)           KO
·130 IF A<1 OR A>6 THEN 30:REM IF A$<A OR
  A$>F THEN MENU                         LI
·135 ON A GOTO 140,310,470,670,850,1010  OM
·136 REM ****** DECIMAL TO HEX *******   GG
·140 PRINT CHR$(147):PRINT "[7" "]DECIMAL
 -TO-HEX CONVERSION"                     OI
·150 PRINT:PRINT "[9" "]RANGE: 0 TO [8"9"
 ]":PRINT                                FA
·160 A$="":PRINT:PRINT "TYPE DECIMAL NUMB
 ER (OR 'M' FOR MENU)":INPUT A$          JB
·165 IF A$="" THEN 160                   DI
·170 FOR L=1 TO 8:HEX$(L)="":NEXT L      EN
·180 IF A$="M" THEN 40                   DO
·190 FOR L=1 TO 8:T$=RIGHT$(A$,L)        DJ
·195 IF ASC(T$)<48 OR ASC(T$)>57 THEN 160 NN
·200 NEXT L                              MO
·205 IF LEN(A$)<1 OR LEN(A$)>8 THEN 160  EH
·210 N=VAL(A$)                           JO
·220 I=8                                 DC
·230 TMP=N:N=INT(N/16)                   KO
·240 TMP=TMP-N*16                        LH
·250 IF TMP<10 THEN HEX$(I)=RIGHT$(STR$(T
 MP),1):GOTO 270                         LC
·260 HEX$(I)=CHR$(TMP-10+ASC("A"))       JF
·270 IF N<>0 THEN I=I-1:GOTO 230         HJ
·280 PRINT "HEX: ";                      GG
·290 FOR L=1 TO 8:PRINT HEX$(L);:NEXT L:P
 RINT                                    BP
·300 GOTO 160                            CH
·305 REM ****** HEX TO DECIMAL ********* JD
·310 PRINT CHR$(147):PRINT "[7" "]HEX-TO-
 DECIMAL CONVERSION"                     LE
·320 PRINT:PRINT "[9" "]RANGE: 0 TO [8"F"
 ]":PRINT                                KI
·330 A$="":PRINT:PRINT "TYPE HEX NUMBER (
 OR 'M'FOR MENU):":INPUT A$              IO
·340 IF A$="M" THEN 40                   DO
·350 IF LEN(A$)>8 THEN 330               EB
·360 N=0                                 DD
·370 FOR L=1 TO LEN(A$)                  OB
·380 HEX$(L)=MID$(A$,L,1)                KP
·385 IF HEX$(L)<"0" OR HEX$(L)>"F" THEN 3
 30                                      CP
·386 IF HEX$(L)>"9" AND HEX$(L)<"A" THEN
 330                                     PA
·390 IF HEX$(L)<="9" THEN N=N*16+VAL(HEX$
 (L)):GOTO 430                           PE
·400 IF HEX$(L)<"A" THEN 330             LL
·410 IF HEX$(L)>"F" THEN 330             MK
·420 N=N*16+ASC(HEX$(L))-ASC("A")+10     BJ
·425 IF N<0 THEN 330                     PK
·430 NEXT L                              MO
·440 PRINT "DEC: ";N:PRINT               IJ
·450 GOTO 330                            CC
·470 REM ***** BINARY TO DECIMAL ******* AH
·490 PRINT CHR$(147):REM CLEAR SCREEN    EG
·500 PRINT:PRINT "[6" "]BINARY-DECIMAL CO
 NVERSION"                               HJ
·510 A$="":PRINT:PRINT "ENTER AN 8-BIT BI
 NARY NUMBER"                            PB
·515 INPUT "(OR 'M' FOR MENU): ";A$:PRINT JM
·517 IF A$="M" THEN 40                   DO
·520 IF LEN(A$)<>8 THEN 510              BL
·530 FOR L=8 TO 1 STEP -1                BB
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
·540 BIT$(L)=MID$(A$,L,1)                    IN
·550 IF BIT$(L)<>"0" AND BIT$(L)<>"1" THE
N 510                                        MA
·560 NEXTL:FORL=1TO8:BIT(L)=VAL(BIT$(L))     ON
·590 NEXTL:ANS=0:M=256:FORL=1TO8             MA
·630 M=M/2:ANS=ANS+BIT(L)*M:NEXTL            BE
·650 PRINT "DECIMAL:";ANS:GOTO510            MP
·670 REM **** DECIMAL TO BINARY *******      GN
·680 PRINT CHR$(147):REM CLEAR SCREEN        EG
·690 PRINT:PRINT "[5" "]DECIMAL-BINARY CO
NVERSION":PRINT                              PG
·700 PRINT "[9" "]RANGE: 0 TO 255":PRINT     GE
·710 PRINT:BM$="":A$="":INPUT"TYPE A NUMB
ER (OR 'M' FOR MENU)";A$:IFA$="M"THEN40      CA
·715 IF A$="0" THEN 840                      EG
·720 IF VAL(A$)<1 OR VAL(A$)>255 THEN 710    EB
·730 NR=VAL(A$):FORL=8TO1STEP-1:Q=NR/2       OK
·760 R=Q-INT(Q)                              FO
·770 IF R=0 THEN BIT$(L)="0":GOTO 790        HJ
·780 BIT$(L)="1"                             NE
·790 NR=INT(Q):NEXTL                         LB
·810 PRINT "BINARY: ";FORL=1TO8:PRINTBIT$
(L);:NEXTL:PRINT:GOTO710                     IA
·840 FOR L=1 TO 8:BIT$(L)="0":NEXT L:GOTO
 810                                         DA
·845 REM ******* HEX TO BINARY ********      KD
·850 PRINT CHR$(147):PRINT "[7" "]HEX-TO-
BINARY CONVERSION":PRINT                     AH
·860 PRINT"[12" "]RANGE: 0 TO FF":PRINT      DM
·870 PRINT:PRINT "TYPE HEX NUMBER (OR 'M'
FOR MENU):":A$="":INPUT A$                   JG
·880 IF A$="M" THEN 40                       DO
·890 IF LEN(A$)>2 OR LEN(A$)<1 THEN 870      EB
·900 HEX$(1)="":HEX$(2)="":FORL=1TOLEN(A$
):HEX$(L)=MID$(A$,L,1)                       MJ
·930 IFHEX$(L)<"0"ORHEX$(L)>"F"THEN870       CE
·940 IFHEX$(L)>"9"ANDHEX$(L)<"A"THEN870      DN
·960 NEXTL:IF HEX$(2)="" THEN HEX$(2)=HEX
$(1):HEX$(1)="0"                             LM
·970 FOR L=1 TO 16:IF HEX$(1)=H$(L) THEN
BIT$(1)=B$(L)                                HC
·980 NEXTL:FOR L=1 TO 16:IF HEX$(2)=H$(L)
 THEN BIT$(2)=B$(L)                          IG
·995 NEXTL:PRINT:PRINT "BIN: ";              EB
·1000 PRINT BIT$(1);BIT$(2):GOTO 870         GF
```

PROGRAMMERS...

We're always interested in looking at top-quality programs for the Commodore home computers. However, a great many factors other than the talent of the programmer contribute toward making a program top-quality. Before submitting a program to *Ahoy!*, we recommend that you send a stamped and self-addressed letter-size envelope to *Ahoy! Writer's and Programmer's Guidelines*, Ion International Inc., 45 West 34th Street—Suite 407, New York, NY 10001.

```
·1010 REM ***** BINARY TO HEX ********        JE
·1020 PRINT CHR$(147):PRINT                   CI
·1030 PRINT"[8" "]BINARY-HEX CONVERSION"      CA
·1040 PRINT:PRINT "ENTER AN 8-BIT BINARY
NUMBER"                                       FL
·1050 A$="":INPUT "(OR 'M' FOR MENU): ";A
$:PRINT:IFA$="M"THEN40                        EF
·1070 IF LEN(A$)<>8 THEN 1040                 DC
·1100 FORL=8TO1STEP-1:BIT$(L)=MID$(A$,L,1
):IFBIT$(L)<>"0"ANDBIT$(L)<>"1"THEN510        KE
·1110 NEXTL:BIT$=BIT$(1)+BIT$(2)+BIT$(3)+
BIT$(4)+BIT$(5)+BIT$(6)+BIT$(7)+BIT$(8)       BG
·1120 T1$=LEFT$(BIT$,4):T2$=RIGHT$(BIT$,4
):FORL=1TO16                                  NM
·1130 IF T1$=B$(L) THEN HEX$(1)=H$(L)         IL
·1150 NEXTL:FOR L=1 TO 16:IF T2$=B$(L) TH
EN HEX$(2)=H$(L)                              NK
·1170 NEXTL:PRINT "HEX: ";HEX$(1);HEX$(2)     LF
·1180 GOTO 1040                               FO
```

R A I D

FROM PAGE 60

CASSETTE BOOT

```
·10 PRINT CHR$(147);CHR$(5):POKE 53280,2:
POKE 53281,0                                  NI
·40 PRINT CHR$(19);"POKE 8192,0:POKE 44,3
2:NEW"                                        GK
·50 PRINT CHR$(17);CHR$(17);"LOAD"CHR$(34
)"RAID"CHR$(34)                               LG
·60 POKE 198,10                               NH
·70 DATA 19,13,13,32,32,32,82,85,78,13        ME
·80 FOR I=1 TO 10:READ X:POKE 630+I,X:NEX
T                                             HM
```

DISK BOOT

```
·10 PRINT CHR$(147);CHR$(5):POKE 53280,2:
POKE 53281,0                                  NI
·20 FOR I=1 TO 6:PRINT CHR$(17):NEXT          LJ
·30 FOR I=1 TO 16:PRINT CHR$(29);:NEXT:PR
INT"LOADING"                                  MN
·40 PRINT CHR$(19);CHR$(31);"POKE 8192,0:
POKE 44,32:NEW"                               CO
·50 PRINT CHR$(17);CHR$(17);"LOAD"CHR$(34
)"RAID"CHR$(34)",8"                           BK
·60 POKE 198,10                               NH
·70 DATA 19,13,13,32,32,32,82,85,78,13        ME
·80 FOR I=1 TO 10:READ X:POKE 630+I,X:NEX
T                                             HM
```

MAIN PROGRAM

```
·1 POKE53280,1:POKE53281,0:POKE646,1          NN
·9 REM    100-MAINLINE                        KE
·10 REM   1000-ONE-TIME INITIALIZATION        AO
·11 REM   1100-VARIABLES FOR BACKGROUND       PH
·12 REM   1200-SETUP CHAR SET                 DM
```

```
•13 REM  1300-DISPLAY SCORE                OO
•14 REM  1400-DISPLAY BACKGROUND           GK
•17 REM  1700-DISPLAY INSTRUCTIONS         HF
•19 REM  1900-GET PLAYER INFO              HO
•21 REM  2100-READ MACHINE LANGUAGE        PA
•22 REM  2200-INIT MACHINE CODE            PO
•25 REM  2500-CRASH SOUND/DISPLAY          MF
•110 GOSUB 1000:REM ONE-TIME INIT          MC
•120 GOSUB 1100:REM VARS FOR BACKGROUND    NA
•130 GOSUB 1700:REM DISPLAY INSTRUCTIONS   AJ
•140 GOSUB 1200:REM SETUP CHAR SET         NF
•300 GOSUB 1900:REM GET PLAYER INFO        CC
•400 GOSUB 1400:REM DISPLAY BACKGROUND     DG
•420 GOSUB 2200:REM INIT MACHINE CODE      NG
•425 JV=PEEK(56320)AND31:IFJV=31THEN425    FF
•430 SYS 49664:IF PEEK(49197)=0 THEN GOSU
 B 2500                                    JJ
•435 HX=0:FOR I=1 TO 29:IF PEEK(CRT+HL(I)
 )=83 OR HX(PL,I)<>0 THEN 445              FI
•440 PT(PL)=PT(PL)+10:HX(PL,I)=1:HX=1      PA
•445 NEXT                                  IA
•450 IF PEEK(49197)=0 THEN LI(PL)=LI(PL)-
 1:GOTO 470                                MK
•460 PT(PL)=PT(PL)+500:LV(PL)=LV(PL)+1     OI
•465 FOR I=1 TO 29:HX(PL,I)=0:NEXT         HJ
•470 IF PEEK(49152)=0 AND HX=1 THEN PT(PL
 )=PT(PL)+100                              GN
•480 GOSUB 1300:REM DISPLAY SCORE          IJ
•485 IF TL=0 THEN 300                      GD
•490 GOTO 400                              CA
•999 END                                   IC
•1000 REM-------ONE-TIME INITIALIZATION    KI
•1005 POKE53280,2:POKE53281,0:POKE646,1    BI
•1010 X=RND(-TI):DIM ML(16),HL(29),HX(9,2
 9):CT=0                                   OP
•1015 FOR I=1 TO 16:READ V                 MP
•1020 IF V<0 THEN GOSUB 50000:GOTO 1030    IK
•1025 ML(I)=V:CT=CT-V                      HL
•1030 NEXT                                 IA
•1035 DATA 123,126,133,136,143,146,153,15
 6,-1116                                   LN
•1040 DATA 843,846,853,856,863,866,873,87
 6,-6876                                   HF
•1045 FOR I=1 TO 29:READ V                 MP
•1050 IF V<0 THEN GOSUB 50000:GOTO 1060    IP
•1055 HL(I)=V:CT=CT-V                      EK
•1060 NEXT                                 IA
•1065 DATA 81,88,91,98,101,108,111,118,-7
 96                                        MI
•1070 DATA 881,888,891,898,901,908,911,91
 8,-7196                                   DH
•1075 DATA 249,250,269,270,729,730,749,75
 0,-3996                                   MP
•1080 DATA 483,499,500,516,517,-2515       EA
•1090 SID=54272                            JK
•1092 CRT=1024:CO=55296:MC(1)=4:MC(2)=12:
 MC(3)=8:MC(4)=3                           DH
```

```
•1094 DEF FNR(X)=INT(RND(1)*X)+1           PC
•1096 SID=54272                            JK
•1098 RETURN                               IM
•1100 REM-------------VARS FOR BACKGROUND  HE
•1110 RV$=CHR$(18):RF$=CHR$(146):W1$=RV$+
 " "+RF$+"[8" "]":W2$=RV$+" "+RF$          FG
•1120 L1$=RV$+"[40" "]"+RF$                MN
•1130 L2$=W1$+W2$+W1$+W2$+W1$+W2$+W1$+W2$ CB
•1140 L3$=W1$+"[10" "]"+W2$+W1$+"[10" "]"
 +W2$                                      AP
•1150 L4$=W2$+W2$+W2$+W2$+"[5" "]"+W2$+W1
 $+W2$+W1$+W2$+W2$+"[5" "]"                AN
•1160 L4$=L4$+W2$+W2$+W2$+W2$              MN
•1170 L5$=W1$+W2$+W1$+"[10" "]"+W2$+W1$+W
 2$                                        HH
•1180 CD$=CHR$(17):CU$=CHR$(145):CL$=CHR$
 (157):CR$=CHR$(29):CS$=CHR$(147)          NK
•1185 HM$=CHR$(19)                         PF
•1199 RETURN                               IM
•1200 REM-------------------COPY CHARSET   GE
•1202 POKE 56334,PEEK(56334)AND254         DN
•1205 POKE 1,PEEK(1)AND251                 IM
•1210 SYS 49667                            KO
•1215 POKE 1,PEEK(1)OR4                    BE
•1220 POKE 56334,PEEK(56334)OR1            PD
•1225 POKE2864,0:POKE2865,231:POKE2866,12
 6:POKE2867,60                             AG
•1230 POKE 2868,24:POKE 2869,0:POKE 2870,
 0:POKE 2871,0                             ND
•1235 POKE 2696,0:POKE 2697,126:POKE 2698
 ,120:POKE 2699,120                        EP
•1240 POKE 2700,27:POKE 2701,27:POKE 2702
 ,120:POKE 2703,0                          EO
•1245 POKE 53272,(PEEK(53272)AND240)OR2    ID
•1299 RETURN                               IM
•1300 REM------------------DISPLAY SCORES  GK
•1302 PRINT CS$;CD$"PLAYER  SCORE  LEVEL
  LIVES LEFT"                              FL
•1305 PRINT"[6"-"]  [5"-"]  [5"-"]  [10"-
 "]"                                       DA
•1310 TL=0:FOR I=1TONP:PL$="  "+STR$(I)+"
 [3" "]"                                   HJ
•1315 SC$=STR$(PT(I))+"  "                 BH
•1320 IFLEN(SC$)<7THENSC$=" "+SC$:GOTO132
 0                                         JJ
•1325 LV$=STR$(LV(I))+"  "                 AP
•1330 IFLEN(LV$)<7THENLV$=" "+LV$:GOTO133
 0                                         AM
•1335 TL=TL+LI(I):LI$="[4" "]"+STR$(LI(I)
 ):IFLI(I)=0THENLI$="[3" "]*DEAD*"         HM
•1340 PRINT PL$;SC$;LV$;LI$:NEXT           OJ
•1345 IF TL=0 THEN 1375                    HK
•1350 PL=PL+1:IF PL>NP THEN PL=1           HE
•1355 IF LI(PL)=0 THEN 1350                GL
•1360 IF NP>1 THEN PRINT CD$"NEXT PLAYER
 IS PLAYER";PL                             NG
•1365 SYS 49673:FOR I=1 TO 500:NEXT        IJ
```

```
·1370 LV=LV(PL):RETURN                          KC
·1375 PRINT CD$;CD$"ANOTHER GAME? ";            EN
·1380 GET X$:IF X$<>"Y"AND X$<>"N"THEN 13
 80                                             CN
·1385 PRINT X$:IF X$="N" THEN END               LF
·1390 RETURN                                    IM
·1400 REM--------------DISPLAY BACKGROUND       LL
·1401 BO%=FNR(16)-1:POKE53280,BO%               OP
·1402 PRINT CS$:POKE53281,0:POKE646,7           LG
·1404 PRINT L1$;:FOR I=1 TO 4:PRINT L2$;:
 NEXT:PRINT L3$;                                GD
·1406 FOR I=1 TO 4:PRINT L2$;:NEXT:PRINT
 L4$;:PRINT L5$;                                CA
·1408 PRINT L4$;:FOR I=1 TO 4:PRINT L2$;:
 NEXT                                           KO
·1410 PRINT L3$;:FOR I=1 TO 4:PRINT L2$;:
 NEXT:PRINT L1$;                                BH
·1420 PRINT"PLAYER";PL;"[7" "]LEVEL";LV;H
 M$                                             IK
·1425 IX=0                                      IK
·1430 FOR I=1 TO 16:IX=IX+1:IF IX>4 THEN
 IX=1                                           PN
·1435 POKE CRT+ML(I),102:POKE CO+ML(I),MC
 (IX):NEXT:NH=0                                 DM
·1440 FOR I=1 TO 29:IF HX(PL,I)<>0 THEN 1
 447                                            AC
·1445 POKE CRT+HL(I),83:POKE CO+HL(I),10:
 NH=NH+1                                        FK
·1447 NEXT                                      IA
·1450 POKE CRT+481,81:POKE CO+481,14            LB
·1499 RETURN                                    IM
·1700 REM------------DISPLAY INSTRUCTIONS       LK
·1702 PRINT CS$;CD$"[17" "]RAID [3"!"]"         JM
·1705 PRINT CD$"[8" "](C) 1983 BY LARRY C
 OATS"                                          IA
·1710 PRINT CD$"COLLECT ALL THE HEARTS WH
 ILE AVOIDING"                                  JF
·1715 PRINT"THE BUTTERFLIES.  THEN EXIT T
 HE SCREEN"                                     FH
·1720 PRINT"ON THE RIGHT."                      HI
·1725 PRINT CD$"SCORING:"                       NP
·1730 PRINT"[3" "]10 POINTS PER HEART"          IC
·1735 PRINT"  100 POINTS FOR GETTING LAST
  HEART"                                        DH
·1740 PRINT"  500 POINTS FOR SAFE EXIT"         CJ
·1745 PRINT CD$;CD$                             BP
·1750 GOSUB 2100:REM LOAD MACHINE LANGUAG
 E                                              EO
·1760 SYS 49670:REM PLAY SONG                   NJ
·1799 RETURN                                    IM
·1900 REM------------------GET PLAYER INFO      DK
·1902 NP=0:PRINT:PRINT CD$"HOW MANY PLAYE
 RS[SS](1-9)? ";                                OB
·1905 GET NP$:IF NP$=""THEN 1905                FO
·1910 IF NP$<"1"OR NP$>"9"THEN 1905             KP
·1915 PRINT NP$:NP=VAL(NP$)                     ML
·1920 PRINT CD$"WHAT SKILL LEVEL (1-9)? "
 ;                                              FK
·1925 GET X$:IF X$=""THEN 1925                  ND
·1950 IF X$<"1"OR X$>"9"THEN 1925               ME
·1955 PRINT X$:LV=VAL(X$)                       DA
·1960 FOR I=1 TO NP:LV(I)=LV:PT(I)=0:LI(I
 )=5                                            KO
·1965 FOR J=1 TO 29:HX(I,J)=0:NEXT              GA
·1970 NEXT                                      IA
·1995 PL=1:PT=0:RETURN                          DK
·2100 REM---------------LOAD MACHINE LANG       HM
·2110 PRINT CD$;"[4" "][3"*"] LOADING MAC
 HINE LANGUAGE [3"*"]";CD$                      CB
·2120 FOR I=49408 TO 50716:READ V               HH
·2130 IF V<0 THEN GOSUB 50000:GOTO 2150         IK
·2140 PRINT TAB(15);I:PRINT CU$;:POKE I,V
 :CT=CT-V                                       EI
·2150 NEXT                                      IA
·2200 REM-----------------INIT ML STORAGE       DB
·2202 POKE49152,NH                              GC
·2205 POKE49154,1505/256:POKE49153,1505-P
 EEK(49154)*256                                 IO
·2210 X=49157:FOR I=1 TO 16:ML=CRT+ML(I)        DO
·2215 POKE X+1,ML/256:POKE X,ML-PEEK(X+1)
 *256:X=X+2:NEXT                                NG
·2220 X=RND(-TI):POKE49193,FNR(256)-1:POK
 E49194,FNR(256)-1                              PH
·2225 POKE49197,0:POKE49216,0                   BC
·2230 DL%=255-12*LV:IF DL%<0 THEN DL%=0         GC
·2235 POKE 49217,DL%                            HN
·2299 RETURN                                    IM
·2500 REM-------------------CRASH EFFECTS       OD
·2510 FORIX=0TO28:POKESID+IX,0:NEXT             DP
·2520 POKE SID+24,15                            AH
·2530 POKESID,20:POKESID+1,30:POKESID+5,0
 :POKESID+6,15*16+9                             EP
·2535 SP%=PEEK(53272)                           IH
·2537 SQ%=FNR(256)-1:IF(SQ%AND240)=(SP%AN
 D240)OR(SQ%AND15)=(SP%AND15)THEN2537           JJ
·2540 POKESID+4,129:POKE53272,SQ%               PK
·2550 FORIX=1TO20:NEXT:POKESID+4,128            AH
·2560 FORIX=1TO3000:NEXT                        CN
·2570 POKE53272,SP%                             IM
·2599 RETURN                                    IM
·20000 A=49408:ZH%=A/256:ZL%=A-(256*ZH%):
 POKE49152,ZH%:POKE49153,ZL%                    HK
·20005 B=30000:ZH%=B/256:ZL%=B-(256*ZH%):
 POKE49154,ZH%:POKE49155,ZL%                    OP
·20010 A=PEEK(49153)+(256*PEEK(49152)):IF
 A=0 THEN END                                   FA
·20015 B=PEEK(49155)+(256*PEEK(49154))          JL
·20020 C=A+15:D=C+1:IF C>=50716 THEN C=50
 716:D=0                                        KF
·20025 ZH%=D/256:ZL%=D-(256*ZH%):POKE 491
 52,ZH%:POKE49153,ZL%                           CI
·20030 D=B+10:ZH%=D/256:ZL%=D-(256*ZH%):P
 OKE 49154,ZH%:POKE49155,ZL%                    GE
·20040 PRINT"[CLEAR][DOWN][DOWN]";B;"DATA
 ";                                             OG
·20050 FOR I=A TO C:I$=STR$(PEEK(I)):J$=R
```

```
IGHT$(I$,LEN(I$)-1)                         OK
·20060 PRINT J$;",";                        OE
·20070 NEXT:PRINT "[LEFT] ":PRINT"GOTO 20
010"                                        PP
·20080 PRINT"[HOME]";                       CE
·20090 POKE 631,13:POKE 632,13:POKE 198,2   HG
·20095 END                                  IC
·21000 REM-----DELETE ML                    NI
·21005 B=30000:ZH%=B/256:ZL%=B-(256*ZH%):
POKE49154,ZH%:POKE49155,ZL%                 OP
·21015 B=PEEK(49155)+(256*PEEK(49154))      JL
·21030 D=B+10:ZH%=D/256:ZL%=D-(256*ZH%):P
OKE 49154,ZH%:POKE49155,ZL%                 GE
·21040 PRINT"[CLEAR][DOWN][DOWN]";B         AM
·21070 PRINT "[LEFT] ":PRINT"GOTO 21015"    EG
·21080 PRINT"[HOME]";                       CE
·21090 POKE 631,13:POKE 632,13:POKE 198,2   HG
·30000 DATA 226,219,98,91,219,215,91,215,
217,215,89,87,215,254,87,254,-2792          CP
·30010 DATA 226,219,98,219,228,219,100,91
,230,224,102,217,231,219,231,219,-3073      HG
·30020 DATA 231,219,103,91,226,217,98,89,
226,219,226,219,226,219,98,91,-2798         FP
·30030 DATA 255,255,226,219,98,91,219,215
,91,215,217,215,89,87,215,254,-2961         GE
·30040 DATA 87,254,226,219,98,219,228,219
,100,91,230,224,102,217,231,219,-2964       PM
·30050 DATA 231,219,231,219,103,91,226,21
7,98,89,226,219,226,219,226,219,-3059       HO
·30060 DATA 98,91,228,224,100,96,226,219,
98,91,224,217,96,89,219,215,-2531           GI
·30070 DATA 91,87,224,217,96,89,219,215,9
1,87,217,214,89,86,219,215,-2456            OH
·30080 DATA 219,215,219,215,91,87,215,210
,87,82,215,210,215,210,215,210,-2915        PD
·30090 DATA 87,82,226,219,98,91,219,215,9
1,215,217,215,89,87,215,254,-2620           NM
·30100 DATA 87,254,226,219,98,91,228,225,
100,97,230,226,102,98,231,228,-2740         MD
·30110 DATA 103,100,230,226,102,98,231,22
8,103,100,228,225,100,97,230,226,-2627      CI
·30120 DATA 102,98,231,228,103,100,230,22
6,102,98,228,225,100,97,226,219,-2613       IN
·30130 DATA 98,91,217,217,89,217,226,217,
98,89,225,215,97,87,226,214,-2623           BN
·30140 DATA 226,214,226,214,98,86,226,214
,98,86,226,214,226,214,226,214,-3008        MP
·30150 DATA 98,86,255,255,255,255,255,255
,255,255,255,255,255,255,255,255,-3754      FJ
·30160 DATA 76,126,194,76,78,196,76,125,1
96,76,142,196,0,39,41,80,-1717              DF
·30170 DATA 255,255,255,255,255,82,2,42,2
55,80,0,40,255,81,1,255,-2368               BI
·30180 DATA 255,255,255,255,255,81,1,255,
255,81,1,255,255,255,255,255,-3224          IC
·30190 DATA 255,255,255,255,255,42,42,255
,255,40,40,255,255,255,255,255,-3224        EP
·30200 DATA 0,231,126,60,24,0,0,0,0,0,0,2
4,60,126,231,0,-882                         AI
·30210 DATA 3,0,0,34,194,0,0,34,194,0,0,3
4,194,0,0,0,-687                            NJ
·30220 DATA 15,32,32,32,255,104,30,24,134
,126,250,6,172,243,230,143,-1828            FA
·30230 DATA 248,46,134,142,150,159,168,17
9,189,200,212,225,238,253,32,188,-2763      LG
·30240 DATA 194,24,173,45,192,109,64,192,
240,1,96,32,38,196,32,130,-1758             JI
·30250 DATA 195,173,64,192,240,1,96,32,38
,196,238,37,192,173,37,192,-2096            AN
·30260 DATA 41,1,10,10,10,168,162,0,185,6
4,194,157,48,11,200,232,-1493               IP
·30270 DATA 224,8,208,244,169,11,141,18,2
12,76,126,194,56,173,1,192,-2053            PF
·30280 DATA 233,41,133,251,133,253,173,2,
192,233,0,133,252,24,105,212,-2370          KE
·30290 DATA 133,254,160,41,140,68,192,173
,0,220,41,15,170,188,16,194,-2005           KJ
·30300 DATA 16,3,76,59,195,177,251,201,10
2,208,4,238,64,192,96,201,-2083             PL
·30310 DATA 83,208,9,206,0,192,32,52,196,
76,3,195,201,32,240,3,-1728                 KK
·30320 DATA 76,76,195,140,68,192,160,41,1
69,32,145,251,177,253,172,68,-2215          IJ
·30330 DATA 192,145,253,169,81,145,251,15
2,24,101,251,141,1,192,165,252,-2515        HK
·30340 DATA 105,0,141,2,192,173,0,192,208
,17,173,1,192,201,6,208,-1811               PO
·30350 DATA 10,173,2,192,201,6,208,3,238,
45,192,172,68,192,177,253,-2132             EL
·30360 DATA 24,105,1,41,15,208,2,169,1,14
5,253,96,173,0,220,41,-1494                 JK
·30370 DATA 15,170,188,32,194,16,3,76,59,
195,177,251,201,83,240,4,-1904              KE
·30380 DATA 201,32,208,3,76,239,194,173,0
,220,41,15,170,188,48,194,-2002             NH
·30390 DATA 16,3,76,59,195,177,251,201,83
,240,4,201,32,208,188,76,-2010              JL
·30400 DATA 239,194,162,30,142,63,192,189
,6,192,141,4,192,189,5,192,-2132            HB
·30410 DATA 141,3,192,32,179,195,174,63,1
92,173,4,192,157,6,192,173,-2068            MG
·30420 DATA 3,192,157,5,192,202,202,142,6
3,192,173,64,192,208,3,138,-2128            PM
·30430 DATA 16,213,96,56,173,3,192,233,40
,133,251,133,253,173,4,192,-2161            PI
·30440 DATA 233,0,133,252,24,105,212,133,
254,169,81,162,3,188,12,194,-2155           LI
·30450 DATA 209,251,240,19,202,16,246,32,
13,196,41,3,170,188,12,194,-2032            JK
·30460 DATA 169,32,209,251,240,4,96,238,6
4,192,140,68,192,160,40,169,-2264           MP
·30470 DATA 32,145,251,177,253,172,68,192
,145,253,169,102,145,251,152,24,-2531       JH
·30480 DATA 101,251,141,3,192,165,252,105
,0,141,4,192,96,173,41,192,-2049            CO
·30490 DATA 42,77,41,192,106,238,42,192,1
```

To enter DOS Plus... you must enter *DOS Boot* in the usual manner, prescribed on pages 85 and 86, then enter the main program by using *Flankspeed* (see page 86).

```
09,42,192,80,3,238,42,192,-1828                   CO
•30500 DATA 141,41,192,41,127,96,174,65,1
92,240,8,160,64,136,208,253,-2138                 EI
•30510 DATA 202,208,248,96,169,20,141,15,
212,169,1,141,19,212,169,200,-2222                HP
•30520 DATA 141,20,212,169,15,141,24,212,
169,17,141,18,212,96,169,0,-1756                  CI
•30530 DATA 133,251,133,253,169,208,133,2
52,169,8,133,254,162,8,160,0,-2426                HF
•30540 DATA 177,251,145,253,200,208,249,2
30,252,230,254,202,208,240,162,21,-3282           ED
•30550 DATA 189,80,194,157,69,192,202,16,
247,32,159,196,96,169,50,141,-2189                ND
•30560 DATA 91,192,169,193,141,92,192,32,
110,196,32,170,197,96,169,0,-2072                 AG
•30570 DATA 141,91,192,169,193,141,92,192
,32,110,196,32,170,197,96,169,-2213               FK
•30580 DATA 4,44,69,192,240,24,173,70,192
,141,2,212,173,71,192,141,-1940                   GJ
•30590 DATA 3,212,173,72,192,141,5,212,17
3,73,192,141,6,212,169,2,-1978                    AE
•30600 DATA 44,69,192,240,24,173,74,192,1
41,9,212,173,75,192,141,10,-1961                  FI
•30610 DATA 212,173,76,192,141,12,212,173
,77,192,141,13,212,169,1,44,-2040                 GN
•30620 DATA 69,192,240,24,173,78,192,141,
16,212,173,79,192,141,17,212,-2151                PD
•30630 DATA 173,80,192,141,19,212,173,81,
192,141,20,212,162,3,189,82,-2072                 KL
•30640 DATA 192,157,21,212,202,16,247,173
,91,192,141,93,192,173,92,192,-2386               MN
•30650 DATA 141,94,192,96,169,4,44,69,192
,240,42,32,211,197,201,255,-2179                  LJ
•30660 DATA 208,1,96,201,254,208,8,169,0,
141,4,212,76,69,197,174,-2018                     MA
•30670 DATA 95,192,142,0,212,174,96,192,1
42,1,212,10,42,41,1,13,-1565                      KE
•30680 DATA 86,192,141,4,212,169,2,44,69,
192,240,42,32,211,197,201,-2034                   DK
•30690 DATA 255,208,1,96,201,254,208,8,16
9,0,141,11,212,76,118,197,-2155                   IM
•30700 DATA 174,95,192,142,7,212,174,96,1
92,142,8,212,10,42,41,1,-1740                     LI
•30710 DATA 13,87,192,141,11,212,169,1,44
,69,192,240,42,32,211,197,-1853                   EF
•30720 DATA 201,255,208,1,96,201,254,208,
8,169,0,141,18,212,76,167,-2215                   CE
•30730 DATA 197,174,95,192,142,14,212,174
,96,192,142,15,212,10,42,41,-1950                 ML
•30740 DATA 1,13,88,192,141,18,212,169,0,
96,173,69,192,41,7,240,-1652                      JM
•30750 DATA 33,173,91,192,141,93,192,173,
92,192,141,94,192,32,20,197,-2048                 BP
•30760 DATA 201,255,240,14,174,90,192,160
,0,136,208,253,202,208,248,76,-2657               KP
•30770 DATA 189,197,96,173,93,192,141,224
,197,173,94,192,141,225,197,173,-2697             FA
•30780 DATA 48,193,238,93,192,208,3,238,9
4,192,201,254,176,3,32,242,-2407                  EE
•30790 DATA 197,96,170,41,15,168,185,102,
194,141,95,192,185,114,194,141,-2230              AA
•30800 DATA 96,192,138,74,74,74,74,24,109
,89,192,41,7,73,7,168,-1432                       LA
•30810 DATA 240,9,78,96,192,110,95,192,13
6,208,247,138,96,-1837                            BL
•50000 IF V<>CT THEN PRINT"ERROR IN DATA
LINE #";PEEK(64)*256+PEEK(63):END                 BI
•50010 I=I-1:CT=0:RETURN                          JG
```

DOS PLUS

FROM PAGE 39

DOS BOOT

```
•40 IFTEM=0THENTEM=49152:LOAD"DOS PLUS",8
,1                                                FD
•41 POKE 53230,001:REM AUTO-VERIFY IF 1           AH
•42 POKE 53012,008:REM DEFAULT DRIVE #            OF
•43 POKE 51329,096:REM UPPER CASE 1525            OA
•44 POKE 53040,018:REM REVERSE VIDEO @$           OI
•45 POKE 53063,018:REM    "[6" "]"    "           DC
•46 POKE 50999,144:REM BLACK HEX #'S              GH
•47 POKE 51069,144:REM    "[4" "]"    "           HJ
•48 POKE 51057,005:REM WHITE ASCII CHRS.          FK
•50 SYSTEM                                        ID
```

DOS PLUS

First byte: C000 Last byte: CFFE

```
C000: A2 02 BD FE CE 95 7C CA 0D
C008: 10 F8 AD 26 03 8D F3 CF 39
C010: AD 27 03 8D F4 CF 78 AD 60
C018: 14 03 8D EF CF AD 15 03 42
C020: 8D F0 CF A9 07 8D 14 03 C3
C028: A9 C1 8D 15 03 58 A9 FF 3B
C030: A0 CD 20 1E AB 68 68 4C A5
C038: 68 CB 8D 9D 03 8E 9C 03 C8
C040: BA BD 01 01 C9 8C D0 16 F7
C048: BD 02 01 C9 A4 D0 0F A2 F9
C050: 00 AD 9D 03 DD 9E CD F0 D9
C058: 16 E8 E0 08 90 F6 AD 9D 13
C060: 03 18 AE 9C 03 C9 3A B0 7E
C068: 03 4C 80 00 4C 8A 00 8D 9C
C070: 9F 03 8E 96 03 AD 14 CF CC
C078: 85 BA 20 88 C6 AE 96 03 70
C080: 8A 0A AA BD 6B CD 48 BD BC
C088: 6A CD 48 60 AD 01 02 F0 0B
C090: 4A C9 3A F0 19 A2 0E BD 57
C098: 9E CD F0 08 CD A7 02 F0 66
C0A0: DF E8 10 F3 A9 1F A0 CE A5
C0A8: 20 1E AB 4C 68 CB A2 08 BD
C0B0: BD 9E CD CD 02 02 F0 C8 66
C0B8: E8 E0 0E 90 F3 B0 E5 A5 51
```

```
C0C0: BA 20 B1 FF A9 6F 85 B9 A5
C0C8: 20 93 FF A0 00 B9 01 02 D9
C0D0: F0 06 20 A8 FF C8 10 F5 5F
C0D8: 20 AE FF AD 14 CF 85 BA 79
C0E0: 20 B4 FF A9 6F 85 B9 20 2E
C0E8: 96 FF A9 35 A0 CE 20 1E 0C
C0F0: AB 20 A5 FF C9 0D F0 06 30
C0F8: 20 D2 FF 4C F1 C0 20 D2 DD
C100: FF 20 AB FF 4C 68 CB 48 94
C108: A5 C5 C9 04 F0 04 C9 05 05
C110: D0 03 8D 9B 03 68 6C EF D4
C118: CF AD A7 02 48 29 0F 8D 4D
C120: 14 CF 85 BA A9 F2 A0 CD 4F
C128: 20 1E AB 68 20 D2 FF 4C B9
C130: 68 CB A9 31 8D C7 CD A9 0C
C138: 00 8D 9E 03 A9 38 8D C8 9F
C140: CD A9 20 8D C9 CD A9 30 D6
C148: 8D CA CD A9 0D 8D CB CD 4C
C150: A9 FF 8D CC CD 20 81 CB 8F
C158: AD 9B 03 F0 08 20 79 C8 FF
C160: A9 0D 20 A8 FF A9 2F A0 59
C168: CF 20 1E AB A2 90 20 85 FA
C170: C4 BD 00 B0 20 D2 FF E8 7F
C178: E0 A2 D0 F5 AD A2 B0 8D 51
C180: 39 CF AD A3 B0 8D 3A CF 23
C188: AD A5 B0 8D 44 CF AD A6 82
C190: B0 8D 45 CF 20 80 C4 A9 F2
C198: 38 A0 CF 20 1E AB 20 85 D0
C1A0: C4 20 A4 C2 20 81 CB AD 08
C1A8: 9B 03 F0 03 20 79 C8 20 BD
C1B0: BD C2 AD 01 B0 C9 FF D0 2B
C1B8: E8 A9 01 8D 15 CF A9 00 68
C1C0: 8D 9E 03 8D A2 03 20 80 C3
C1C8: C4 20 CC FF 20 D7 C1 A9 DD
C1D0: 00 8D 9B 03 4C DB C0 20 06
C1D8: C7 C6 A9 97 85 BB A9 CE 62
C1E0: 85 BC A9 05 85 B7 AD 14 D0
C1E8: CF 85 BA A9 60 85 B9 20 62
C1F0: D5 F3 A5 BA 20 B4 FF A5 95
C1F8: B9 20 96 FF A9 00 85 90 29
C200: A0 22 8C 9F 03 20 A5 FF B7
C208: A4 90 D0 69 AC 9F 03 88 4F
C210: D0 F0 20 A5 FF 48 20 A5 A5
C218: FF 48 AD 9B 03 F0 3E 68 44
C220: 8D B2 02 68 8D B1 02 A2 AE
C228: 00 20 A5 FF F0 06 9D B3 36
C230: 02 E8 D0 F5 20 75 C2 20 5A
C238: 79 C8 AD B2 02 AE B1 02 3F
C240: 20 B0 CC A9 20 20 D2 FF 9A
C248: A2 00 BD B3 02 20 D2 FF 51
C250: E8 E0 0C 90 F5 A9 0D 20 83
C258: A8 FF 4C 78 C2 68 A8 68 02
C260: AA 98 20 B0 CC A9 20 20 2B
C268: D2 FF 20 A5 FF F0 06 20 18
C270: D2 FF 4C 6A C2 20 42 F6 16
C278: 20 E7 FF 60 20 E1 FF F0 D3
C280: 13 20 E4 FF F0 0D C9 20 80
C288: D0 09 20 E4 FF F0 FB C9 1E
C290: 20 F0 F7 60 68 68 68 68 9B
C298: 4C B9 C1 A9 20 20 D2 FF 1D
C2A0: CA D0 F8 60 AE 9E 03 A0 86
C2A8: 00 BD 79 CF 30 07 99 C7 48
C2B0: CD C8 E8 D0 F4 99 C7 CD 25
C2B8: E8 8E 9E 03 60 A0 00 B9 8C
C2C0: 71 CF 8C 9F 03 48 38 20 D1
C2C8: F0 FF A0 01 18 20 F0 FF 84
C2D0: 68 AA 20 F6 C2 A9 00 8D F4
C2D8: A2 03 20 7C C2 AC 9F 03 2D
C2E0: C8 C0 08 90 DA 60 48 AD 34
C2E8: A2 03 D0 08 A9 22 8D A2 63
C2F0: 03 20 D2 FF 68 60 8E A5 E3
C2F8: 03 A0 00 BD 03 B0 F0 15 14
C300: C9 A0 F0 11 99 A7 02 20 CF
C308: E6 C2 C8 20 D2 FF EE A4 01
C310: 03 E8 D0 E7 60 AD A4 03 6A
C318: F0 FA A9 22 20 D2 FF A9 6C
C320: 00 8D A4 03 84 B7 38 20 E9
C328: F0 FF 18 A0 14 20 F0 FF F6
C330: AD 9B 03 F0 08 20 79 C8 D7
C338: A9 00 20 39 C4 AE A5 03 57
C340: BD 00 B0 20 48 C4 38 20 34
C348: F0 FF A0 18 18 20 F0 FF 1B
C350: AD 9B 03 F0 05 A9 04 20 60
C358: 39 C4 AE A5 03 BD 1B B0 37
C360: 48 BD 1C B0 AA 68 A0 00 E6
C368: 8C A9 03 20 B0 CC AE A5 93
C370: 03 BD 01 B0 C9 24 90 03 64
C378: 4C 38 C4 AA A9 00 8D A9 4D
C380: 03 8D 98 03 8D 15 CF 20 3F
C388: B0 CC A2 00 BD 75 03 9D 7C
C390: C7 CD E8 EC 98 03 90 F4 1D
C398: A9 20 9D C7 CD E8 8E 9A A7
C3A0: 03 AE A5 03 BD 02 B0 C9 35
C3A8: 16 90 03 4C 38 C4 AA A9 EF
C3B0: 00 8D 98 03 20 B0 CC A2 1A
C3B8: 00 AC 9A 03 BD 75 03 99 D2
C3C0: C7 CD C8 E8 EC 98 03 90 21
C3C8: F3 A9 20 99 C7 CD C8 A9 28
C3D0: FF 99 C7 CD A9 00 8D CD 05
C3D8: CB A9 B2 8D CE CB A9 05 D7
C3E0: 8D D1 CB 20 81 CB A9 00 23
C3E8: 8D D1 CB A9 00 8D CD CB E4
C3F0: A9 B0 8D CE CB AD 9B 03 BF
C3F8: F0 03 20 79 C8 38 20 F0 98
C400: FF A0 1B 18 20 F0 FF AD 92
C408: 9B 03 F0 05 A9 07 20 39 A6
C410: C4 A9 20 20 D2 FF A9 24 5F
C418: 8D 15 CF 20 D2 FF AE 02 2E
C420: B2 AD 03 B2 20 12 CD A9 DF
C428: 20 20 D2 FF AE 02 B2 AD 4C
C430: 03 B2 20 B0 CC 20 C7 C6 32
C438: 60 48 A9 10 20 A8 FF A9 0D
C440: 02 20 A8 FF 68 4C A8 FF 68
C448: AA 20 80 C4 8A 29 0F 18 33
C450: F0 12 C9 01 F0 14 C9 02 EE
C458: F0 16 C9 03 F0 18 A9 16 F4
```

```
C460: A0 CF B0 16 A9 1B A0 CF CC
C468: 90 10 A9 2A A0 CF B0 0A 08
C470: A9 20 A0 CF B0 04 A9 25 2E
C478: A0 CF 20 1E AB 4C 85 C4 69
C480: A9 37 85 01 60 A9 36 85 AD
C488: 01 60 A6 2B A4 2C AD 9F D9
C490: 03 C9 25 D0 03 A9 01 2C 2D
C498: A9 00 85 B9 A9 00 20 D5 21
C4A0: FF B0 5E AD 9F 03 C9 25 EE
C4A8: F0 57 A5 AF 85 2E A5 AE 4E
C4B0: 85 2D 20 59 A6 20 33 A5 7C
C4B8: AD 9F 03 C9 2F F0 42 A9 DE
C4C0: 00 20 90 FF 20 8E A6 4C 13
C4C8: AE A7 A2 02 BD AB E3 95 A6
C4D0: 7C CA 10 F8 A9 64 A0 A3 73
C4D8: 20 1E AB 78 AD EF CF 8D 36
C4E0: 14 03 AD F0 CF 8D 15 03 0C
C4E8: 58 4C 68 CB 20 59 E1 AD CA
C4F0: EE CF F0 0D A5 2B 8D AA B6
C4F8: 03 A5 2C 8D AB 03 4C ED 44
C500: C8 4C DB C0 A9 E3 A0 CE AE
C508: 20 1E AB A2 00 A9 A7 A0 86
C510: 02 20 1E AB A9 D7 A0 CE EC
C518: 20 1E AB 20 E4 FF F0 FB F3
C520: 8D 9A 03 20 D2 FF A5 2D 11
C528: 48 A5 2E 48 E6 2E A9 00 4B
C530: 85 2D AA A8 A9 30 9D 00 AD
C538: 02 E8 A9 3A 9D 00 02 E8 8F
C540: B9 A7 02 F0 07 9D 00 02 3B
C548: E8 C8 D0 F4 9D 00 02 86 E5
C550: B7 A9 00 85 BB A9 02 85 24
C558: BC AD 14 CF 85 BA A9 62 F2
C560: 85 B9 20 D5 F3 A5 BA 20 0A
C568: B4 FF A5 B9 20 96 FF A0 D3
C570: 00 84 90 20 A5 FF A6 90 82
C578: D0 10 91 2D C8 D0 F4 E6 8D
C580: 2E A5 2E C5 38 90 E8 4C 46
C588: 75 C6 91 2D C8 D0 02 E6 06
C590: 2E 91 2D A5 2E 8D A0 03 82
C598: 8C 9F 03 68 48 85 2E E6 13
C5A0: 2E 20 42 F6 20 E7 FF A9 D9
C5A8: AB A0 CE 20 1E AB 20 E1 AF
C5B0: FF D0 03 4C 7F C6 20 E4 1C
C5B8: FF C9 0D D0 F1 AD 9A 03 9D
C5C0: 29 0F CD 14 CF D0 06 AD 2F
C5C8: 14 CF 8D 9A 03 AD 9A 03 23
C5D0: 29 0F 8D 14 CF 85 BA AA 65
C5D8: A9 08 A0 01 20 BA FF A9 B0
C5E0: 8E A0 CE 20 1E AB A5 B7 26
C5E8: A2 00 A0 02 20 BD FF 20 2C
C5F0: C0 FF AD 14 CF 20 B1 FF 15
C5F8: A9 61 85 B9 20 93 FF A0 97
C600: FF C8 B1 2D 20 A8 FF A5 16
C608: 90 D0 14 CC 9F 03 D0 07 C4
C610: A5 2E CD A0 03 F0 08 C0 0F
C618: FF D0 E6 E6 2E D0 E0 A9 40
C620: 08 20 C3 FF 20 E7 FF 68 7C
C628: 85 2E 68 85 2D 20 C7 C6 A5
C630: E6 2E AD A0 03 38 E5 2E E2
C638: C6 2E AE 9F 03 20 B0 CC 1C
C640: A9 EF A0 CE 20 1E AB AD E0
C648: EE CF F0 26 A2 00 BD 00 7E
C650: 02 F0 06 9D A7 02 E8 10 89
C658: F5 86 B7 A9 A7 85 BB A9 C8
C660: 02 85 BC A5 2E AA E8 A9 B5
C668: 02 8D AA 03 8E AB 03 4C 2F
C670: ED C8 4C DB C0 20 42 F6 69
C678: A9 9C A0 CE 20 1E AB 68 80
C680: 85 2E 68 85 2D 4C DB C0 38
C688: A0 00 A2 00 E8 BD 00 02 74
C690: F0 2A C9 3A F0 02 D0 0B 7E
C698: E8 BD 00 02 D0 05 CA 8A 6C
C6A0: A8 D0 19 C9 22 F0 05 9D B2
C6A8: A6 02 D0 EC E8 BD 00 02 B7
C6B0: F0 0A C9 22 F0 06 99 A7 CF
C6B8: 02 C8 D0 F0 84 B7 A9 A7 D2
C6C0: 85 BB A9 02 85 BC 60 A9 F9
C6C8: 0D 4C D2 FF A0 03 B9 00 52
C6D0: 02 F0 20 C9 2C F0 0E C9 A2
C6D8: 30 90 15 C9 3A B0 11 99 0E
C6E0: A4 02 C8 10 E9 8D 95 03 70
C6E8: A9 20 99 A4 02 4C E2 C6 E8
C6F0: 4C 6F C8 AD 95 03 D0 03 8F
C6F8: 4C 6F C8 A2 00 8A 99 A4 E8
C700: 02 BD A7 02 F0 0C 9D C7 CB
C708: CD 9D D5 CD 9D E2 CD E8 4E
C710: 10 EF A9 0D 9D C7 CD 9D 97
C718: D5 CD 9D E2 CD E8 A9 FF 9C
C720: 9D C7 CD 9D D5 CD 9D E2 15
C728: CD A9 93 20 D2 FF 20 81 C7
C730: CB A9 91 20 D2 FF A9 90 64
C738: 20 D2 FF 20 85 C4 AD 9B DE
C740: 03 F0 08 20 79 C8 A9 0D 55
C748: 20 A8 FF A2 00 A0 00 B9 0E
C750: 00 B0 20 16 CD B9 00 B0 6F
C758: 20 67 C7 C8 F0 06 20 93 1B
C760: C7 4C 4F C7 4C C3 C7 C9 2D
C768: 23 90 16 C9 7F B0 19 48 8D
C770: A9 05 20 D2 FF 68 20 AC 47
C778: C7 20 D2 FF A9 90 4C D2 8C
C780: FF A9 2E 20 AC C7 D0 F6 B4
C788: C9 C0 B0 F5 C9 A1 90 F1 A7
C790: 4C 6F C7 98 A2 00 DD 01 2E
C798: CF F0 06 E8 E0 13 90 F6 C3
C7A0: 60 A9 2E 20 D2 FF A9 0D 82
C7A8: 20 A8 FF 60 48 AD 9B 03 66
C7B0: F0 0F A9 20 20 A8 FF 68 AB
C7B8: 20 A8 FF A9 20 20 A8 FF 14
C7C0: 68 68 60 A2 00 BD 57 CE 78
C7C8: F0 06 20 D2 FF E8 D0 F5 62
C7D0: A2 00 BD C7 CD C9 0D F0 8E
C7D8: 08 20 D2 FF E8 E0 06 90 34
C7E0: F1 A2 00 BD 63 CE F0 06 5C
C7E8: 20 D2 FF E8 10 F5 A2 00 6D
C7F0: A0 00 8A BE 00 B0 20 B0 5C
C7F8: CC A9 20 20 D2 FF A0 01 24
```

```
C800: A9 00 BE 00 B0 20 B0 CC B6
C808: A2 00 BD 6F CE F0 06 20 BD
C810: D2 FF E8 10 F5 AD 9B 03 1E
C818: F0 13 A9 0D 20 A8 FF 20 BB
C820: AE FF 20 E7 FF A9 00 8D 0E
C828: 9B 03 4C 4B C7 78 20 80 3F
C830: C4 AD 14 03 8D F1 CF AD B6
C838: 15 03 8D F2 CF A9 5E 8D 36
C840: 14 03 A9 C8 8D 15 03 58 C7
C848: 20 CF FF C9 0D D0 F9 78 52
C850: A9 07 8D 14 03 A9 C1 8D 9E
C858: 15 03 58 4C E6 C9 38 20 1E
C860: F0 FF E0 14 90 06 A2 00 7F
C868: 18 20 F0 FF 6C F1 CF A9 69
C870: 3E A0 CE 20 1E AB 4C 68 BC
C878: CB A9 82 AE 9B 03 F0 12 C0
C880: A0 60 20 BA FF A9 00 20 26
C888: BD FF 20 C0 FF A2 82 20 6C
C890: C9 FF 60 A2 00 A0 00 A9 A7
C898: 30 99 A7 02 C8 A9 3A 99 52
C8A0: A7 02 C8 BD 03 02 C9 27 C6
C8A8: F0 06 E8 10 F6 4C 6F C8 14
C8B0: E8 BD 03 02 C9 27 F0 0B 49
C8B8: 99 A7 02 C8 C0 27 90 F0 2E
C8C0: 4C 6F C8 20 BC C6 E8 BD 8F
C8C8: 03 02 C9 24 F0 06 E8 10 AB
C8D0: F6 4C 6F C8 A0 00 E8 BD 93
C8D8: 03 02 F0 06 99 75 03 C8 AF
C8E0: D0 F4 A9 00 99 75 03 8E F0
C8E8: 96 03 20 2C CD AD 14 CF 2E
C8F0: 85 BA A9 00 85 B9 A5 B7 77
C8F8: A2 A7 A0 02 20 BD FF A9 6D
C900: 01 AE AA 03 AC AB 03 20 D8
C908: D5 FF A5 90 29 10 F0 07 45
C910: A9 69 A0 A3 20 1E AB 4C 9D
C918: DB C0 A2 00 A0 00 A9 30 D1
C920: 99 A7 02 C8 A9 3A 99 A7 51
C928: 02 C8 BD 03 02 C9 27 F0 97
C930: 06 E8 10 F6 4C 6F C8 E8 93
C938: BD 03 02 C9 27 F0 06 99 7C
C940: A7 02 C8 10 F2 84 B7 A0 92
C948: 00 E8 BD 03 02 C9 24 F0 D2
C950: 06 E8 10 F6 4C A4 C0 E8 E0
C958: BD 03 02 C9 20 F0 0E C9 CD
C960: 2C F0 0A C9 24 F0 06 99 06
C968: 75 03 C8 10 EA A9 00 99 E7
C970: 75 03 8E 96 03 20 2C CD 2B
C978: AD AA 03 8D 9D 03 AD AB 5B
C980: 03 8D 9C 03 AE 96 03 A0 99
C988: 00 BD 03 02 C9 24 F0 06 30
C990: E8 10 F6 4C A4 C0 E8 BD D8
C998: 03 02 F0 06 99 75 03 C8 6F
C9A0: 10 F4 A9 00 99 75 03 20 81
C9A8: 2C CD AD 9D 03 AE 9C 03 3F
C9B0: 85 FB 86 FC AD 14 CF 85 CC
C9B8: BA A9 00 85 B9 A5 B7 A2 5C
C9C0: A7 A0 02 20 BD FF AE AA 42
C9C8: 03 AC AB 03 A9 FB 20 D8 C5
C9D0: FF AD EE CF F0 0D A5 FB DC
C9D8: 8D AA 03 A5 FC 8D AB 03 F2
C9E0: 4C ED C8 4C DB C0 A9 00 76
C9E8: 85 B7 85 BB 85 BC AA 9D F1
C9F0: A7 02 E8 E0 28 90 F8 8D A3
C9F8: A0 03 A9 04 8D 0D CA A9 59
CA00: 00 8D 0C CA 20 85 C4 A2 71
CA08: 00 A0 00 BD 00 04 C9 20 54
CA10: F0 43 C9 2E D0 03 4C 77 D3
CA18: CA C9 30 B0 03 4C 63 CA 0B
CA20: C9 3A 90 03 4C 5B CB 99 C4
CA28: AA 03 C8 C0 02 F0 03 4C A1
CA30: 77 CA AD AA 03 29 0F 0A 10
CA38: 0A 0A 0A 8D AA 03 AD AB EA
CA40: 03 29 0F 18 6D AA 03 8E 3D
CA48: 97 03 A0 00 AE A0 03 9D 73
CA50: 00 B2 E8 D0 03 4C 8A CA 61
CA58: 8E A0 03 AE 97 03 E8 F0 AD
CA60: 22 D0 14 C9 07 90 03 4C 18
CA68: 5B CB C9 01 B0 03 4C 5B B5
CA70: CB 18 69 09 4C 27 CA E8 ED
CA78: F0 03 4C 0B CA EE 0D CA 55
CA80: 4C 0B CA EE 0D CA E8 4C 9E
CA88: 0B CA A2 00 BD 00 B0 DD 4D
CA90: 00 B2 D0 0E E8 D0 F5 A9 7B
CA98: 93 20 D2 FF 20 80 C4 4C D0
CAA0: DB C0 20 80 C4 A9 8B A0 78
CAA8: CE 20 1E AB 20 85 C4 78 44
CAB0: 20 E7 FF A9 0F AE 14 CF 04
CAB8: A0 0F 20 BA FF A9 00 20 0D
CAC0: BD FF 20 C0 FF AD 14 CF F0
CAC8: 20 B1 FF A5 B9 09 60 20 83
CAD0: 93 FF A9 02 AE 14 CF A0 43
CAD8: 02 20 BA FF A9 01 A2 BF C2
CAE0: A0 CD 20 BD FF 20 C0 FF 0E
CAE8: A2 0F 20 C9 FF A2 00 BD E4
CAF0: DC CD 30 06 20 CA F1 E8 97
CAF8: D0 F5 20 CC FF A2 0F 20 7E
CB00: C9 FF A2 00 BD E9 CD 30 12
CB08: 06 20 CA F1 E8 D0 F5 20 BA
CB10: CC FF A2 0F 20 C9 FF A2 1B
CB18: 00 BD CE CD 30 06 20 CA 93
CB20: F1 E8 D0 F5 A2 02 20 C9 50
CB28: FF A2 00 BD 00 B2 20 CA 26
CB30: F1 E8 D0 F7 20 CC FF A9 6A
CB38: 02 20 C3 FF A2 0F 20 C9 B9
CB40: FF A2 00 BD BD CD 20 CA 17
CB48: F1 E8 E0 03 D0 F5 20 CC BA
CB50: FF 20 DC CB 20 80 C4 58 D6
CB58: 4C DB C0 20 80 C4 A9 80 D0
CB60: A0 CE 20 1E AB 4C DB C0 A2
CB68: A2 00 8A 9D 00 02 9D A7 7A
CB70: 02 E8 E0 28 90 F5 A2 00 8D
CB78: 9D 75 03 E8 10 FA 4C 86 55
CB80: E3 78 20 E7 FF A9 0F AE 4C
CB88: 14 CF A0 0F 20 BA FF A9 A0
CB90: 00 20 BD FF 20 C0 FF A9 F8
CB98: 02 AE 14 CF A0 02 20 BA AA
```

```
CBA0: FF A9 01 A2 BF A0 CD 20 3C
CBA8: BD FF 20 C0 FF A2 0F 20 19
CBB0: C9 FF A2 00 BD C0 CD 30 99
CBB8: 06 20 CA F1 E8 D0 F5 20 6B
CBC0: CC FF A2 02 20 C6 FF A2 BB
CBC8: 00 20 A5 FF 9D 00 B0 E8 C5
CBD0: E0 00 D0 F5 A9 02 20 C3 08
CBD8: FF 20 CC FF A2 0F 20 C9 61
CBE0: FF A2 00 BD BB CD 20 CA B5
CBE8: F1 E8 E0 03 D0 F5 20 CC 5B
CBF0: FF A2 0F 20 C9 FF A2 00 2F
CBF8: BD B9 CD 20 CA F1 E8 E0 E4
CC00: 03 D0 F5 20 CC FF 20 E7 BE
CC08: FF 58 60 A2 00 BD 01 02 24
CC10: F0 06 9D 75 03 E8 10 F5 0C
CC18: 20 2C CD A9 3D 20 D2 FF 0C
CC20: AD AB 03 AE AA 03 20 B0 A9
CC28: CC 4C 68 CB A2 07 A9 00 C8
CC30: 9D 75 03 9D 7F 03 9D 8A 8E
CC38: 03 CA 10 F4 A9 3D 20 D2 E4
CC40: FF A9 24 20 D2 FF A2 00 A3
CC48: BD 01 02 F0 57 E8 8E 97 60
CC50: 03 20 59 CC AE 97 03 D0 B3
CC58: EF C9 30 90 3C C9 3A B0 C3
CC60: 38 29 0F A8 20 9A CC B0 B1
CC68: 30 20 9A CC B0 2B A2 07 A5
CC70: 18 BD 7F 03 7D 8A 03 9D 71
CC78: 7F 03 CA 10 F4 B0 1A 20 B5
CC80: 9A CC B0 15 A2 07 98 18 08
CC88: 7D 7F 03 9D 7F 03 9D 8A D0
CC90: 03 A9 00 CA 10 F2 B0 01 BC
CC98: 18 60 A2 07 18 3E 7F 03 93
CCA0: CA 10 FA 60 AE 86 03 AD BC
CCA8: 85 03 20 12 CD 4C 68 CB B1
CCB0: 8D A7 03 8E A6 03 A2 09 CC
CCB8: 8E A8 03 A0 B0 AD A6 03 9B
CCC0: DD 5F CD AD A7 03 FD 60 82
CCC8: CD 90 0F 8D A7 03 AD A6 C2
CCD0: 03 FD 5F CD 8D A6 03 C8 FE
CCD8: D0 E3 98 CA F0 11 C9 B0 6D
CCE0: F0 03 8D A8 03 2C A8 03 E5
CCE8: 30 05 AD A9 03 F0 1F 29 B1
CCF0: 7F 8E 99 03 AE 15 CF D0 FF
CCF8: 0F AE 98 03 9D 75 03 E8 51
CD00: 8E 98 03 AE 99 03 10 06 8B
CD08: AE 99 03 20 D2 FF CA 10 21
CD10: AA 60 20 16 CD 8A 48 4A 3C
CD18: 4A 4A 4A 20 21 CD 68 29 97
CD20: 0F 09 30 C9 3A 90 02 69 68
CD28: 06 4C D2 FF A9 00 8D AB 30
CD30: 03 AA F0 03 0D AA 03 8D 1A
CD38: AA 03 BD 75 03 F0 20 C9 F6
CD40: 30 90 1C 49 30 C9 0A 90 FA
CD48: 08 69 88 C9 FA 90 10 49 F0
CD50: F0 E8 A0 04 0E AA 03 2E B8
CD58: AB 03 88 D0 F7 F0 D5 60 7F
CD60: 01 00 0A 00 64 00 E8 03 BB
CD68: 10 27 0A CC 2B CC 89 C4 BC
CD70: 89 C4 89 C4 8B C0 EB C4 0A
CD78: 03 C5 CB C6 F7 CF FA CF 66
CD80: 92 C8 19 C9 F4 CF 31 C1 76
CD88: C9 C4 18 C1 18 C1 BE C0 4A
CD90: BE C0 BE C0 BE C0 BE C0 8E
CD98: BE C0 BE C0 BE C0 24 23 5E
CDA0: 5E 2F 25 40 5F 2A 44 32 93
CDA8: 33 56 5F 31 24 51 38 39 A9
CDB0: 4E 43 52 53 49 56 4D 55 2A
CDB8: 00 49 30 55 3B 56 30 23 6C
CDC0: 55 31 3A 32 20 30 20 31 55
CDC8: 38 20 31 0D FF FF 55 32 E6
CDD0: 3A 32 20 30 20 31 31 20 30
CDD8: 31 31 0D FF 42 2D 46 3A 38
CDE0: 30 20 31 20 31 0D FF FF C0
CDE8: FF 42 2D 50 3A 32 20 30 65
CDF0: 0D FF 0D 4C 4F 47 47 45 7A
CDF8: 44 20 54 4F 20 23 00 0D 51
CE00: 20 20 20 20 20 20 20 20 01
CE08: 2D 2A 2D 20 44 4F 53 20 B3
CE10: 50 4C 55 53 20 56 31 2E 2B
CE18: 30 20 2D 2A 2D 0D 00 20 1A
CE20: 20 5E 20 42 41 44 20 43 E9
CE28: 4D 44 20 4F 52 20 4E 4F 39
CE30: 20 28 3A 29 00 0D 53 54 90
CE38: 41 54 55 53 3A 00 20 20 F0
CE40: 20 5E 20 42 41 44 20 23 E9
CE48: 20 4F 52 20 4E 4F 20 28 10
CE50: 2C 20 24 20 27 29 00 0D 3E
CE58: 0D 44 55 4D 50 49 4E 47 7B
CE60: 3A 20 00 20 4E 45 58 54 1B
CE68: 20 54 26 53 3A 20 00 0D BD
CE70: 52 45 54 55 52 4E 20 54 C6
CE78: 4F 20 45 58 49 54 13 00 36
CE80: 93 4E 4F 4E 2D 48 45 58 13
CE88: 20 23 00 93 52 45 57 52 A0
CE90: 49 54 49 4E 47 0D 00 24 3E
CE98: 30 3A 3C 3E 0D 46 49 4C 66
CEA0: 45 20 54 4F 4F 20 42 49 A4
CEA8: 47 0D 00 0D 49 4E 53 45 3A
CEB0: 52 54 20 12 44 45 53 54 BA
CEB8: 49 4E 41 54 49 4F 4E 92 5F
CEC0: 20 44 49 53 4B 20 2D 50 AA
CEC8: 52 45 53 53 20 12 52 45 D0
CED0: 54 55 52 4E 92 0D 00 0D C7
CED8: 92 54 4F 20 3F 20 38 2F F5
CEE0: 39 20 00 93 43 4F 50 59 0A
CEE8: 49 4E 47 3A 20 12 00 20 54
CEF0: 42 59 54 45 53 20 43 4F 2C
CEF8: 50 49 45 44 0D 00 4C 3A AF
CF00: C0 0D 1A 27 34 41 4E 5B 2E
CF08: 68 75 82 8F 9C A9 B6 C3 B8
CF10: D0 DD EA F7 08 01 52 45 42
CF18: 4C 20 00 44 45 4C 20 00 7A
CF20: 50 52 47 20 00 55 53 52 25
CF28: 20 00 53 45 51 20 00 93 E5
CF30: 12 44 49 53 4B 3A 20 00 C8
CF38: 2C 20 20 20 44 4F 53 20 CB
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
CF40: 56 45 52 20 20 20 0D 12 AD
CF48: 46 49 4C 45 4E 41 4D 45 8B
CF50: 20 20 20 20 20 20 20 20 51
CF58: 20 20 20 20 54 59 50 20 F6
CF60: 42 4C 4B 20 4C 4F 41 44 7B
CF68: 20 41 44 44 52 53 92 0D 97
CF70: 00 02 22 42 62 82 A2 C2 21
CF78: E2 31 38 20 31 0D FF 31 54
CF80: 38 20 34 0D FF 31 38 20 A3
CF88: 37 0D FF 31 38 20 31 30 B7
CF90: 0D FF 31 38 20 31 33 0D 98
CF98: FF 31 38 20 31 36 0D FF 96
CFA0: 31 38 20 32 0D FF 31 38 D2
CFA8: 20 35 0D FF 31 38 20 38 CC
CFB0: 0D FF 31 38 20 31 31 0D B6
CFB8: FF 31 38 20 31 34 0D FF B4
CFC0: 31 38 20 31 37 0D FF 31 F0
CFC8: 38 20 33 0D FF 31 38 20 EA
CFD0: 36 0D FF 31 38 20 39 0D E3
CFD8: FF 31 38 20 31 32 0D FF D2
CFE0: 31 38 20 31 35 0D FF 31 0F
CFE8: 38 20 31 38 0D FF 01 00 B8
CFF0: 00 00 00 00 00 4C 68 CB 71
CFF8: 4C 68 CB 4C 68 CB 00 F9
```

TWO-COLUMN DIRECTORY

FROM PAGE 30 MAIN PROGRAM

```
•5 REM ****************                          GD
•10 REM DEFINE VARIABLES                         AC
•15 REM ****************                         GD
•20 POKE 53280,6: POKE 53281,0: POKE 646,
 1: REM SET SCREEN, TEXT COLORS                  BJ
•25 COL=1: DIM FI$(40)                           AF
•30 WH$=CHR$(5): GR$=CHR$(30): YE$=CHR$(1
 58): OG$=CHR$(129): PU$=CHR$(156)               LE
•35 CL$=CHR$(147): HO$=CHR$(19): RV$=CHR$
 (145)+CHR$(18): UP$=CHR$(145)                   HN
•40 SP$=" ": FOR BL=0 TO 18: SP$=SP$+" "
 : NEXT                                          KB
•45 OPEN 6,8,0,"$": REM OPEN DIRECTORY           MD
•50 REM **************************               OG
•55 REM FIND AND PRINT DISK HEADER               ND
•60 REM **************************               OG
•65 PRINT CL$: REM CLEAR SCREEN                  BE
•70 GOSUB 535: PRINT HO$;                        PD
•75 GET#6, BY$: IF BY$ <> CHR$(34) THEN 7
 5                                               IN
•80 PRINT SPC(8) "[RVSON]";OG$;BY$;              CE
•85 GET#6, BY$: IF BY$ <> "" THEN PRINT B
 Y$;: GOTO 85                                    GN
•90 PRINT WH$                                    FP
•95 REM  ************************                JL
•100 REM FIND AND PRINT FILENAMES                MM
•105 REM ************************                JL
•110 FOR ENTRY=0 TO 19: I=I+1: F$="": SK=
 0                                               NB
•115 GET#6, BY$,BY$:GET#6, HB$,LB$               BL
•120 GET#6, BY$: IF BY$ <> CHR$(32) THEN
 230                                             FL
•125 GET#6, BY$: IF BY$ <> CHR$(34) THEN
 125                                             IM
•130 IF COL/2=INT(COL/2) THEN PRINT TAB(1
 9)" ";                                          MJ
•135 FOR BL=1 TO 16: GET#6, BY$                  CF
•140 IF BY$="" THEN 160                          KH
•145 IF BY$=CHR$(34) THEN SK=1: GOTO 160         AA
•150 IF SK=0 THEN F$=F$+BY$                      KM
•155 GOTO 165                                    CK
•160 BY$=CHR$(32)                                AA
•165 PRINT BY$;: NEXT: FI$(I)=F$                 AG
•170 GET#6, BY$                                  PM
•175 GET#6, BY$: IF BY$=CHR$(32) THEN 175        MC
•180 PRINT ","; BY$; " "                         JM
•185 GET#6, BY$: IF BY$ <> "" THEN 185           CF
•190 NEXT: COL=COL + 1                           BC
•200 IFCOL > 2 AND COL/2 <> INT(COL/2) TH
 EN I=0:GOSUB 560                                CM
•205 IF COL=2 THEN PRINT HO$: GOTO 110           IA
•210 POKE 214,22: POKE211,0:PRINT RV$;YE$
 ;"  ":GOSUB430: GOSUB 265: PRINT HO$;WH$        OD
•215 IF COL/2 <> INT(COL/2) THEN FOR BL=0
  TO 19: PRINT SP$: NEXT: GOTO 225               EL
•220 FOR BL=0 TO 19: PRINT TAB(19);SP$: N
 EXT: GOSUB 580                                  AG
•225 PRINT SP$;SP$;HO$: GOTO 110                 LL
•230 REM *****************                       JN
•235 REM PRINT BLOCKS FREE                       PD
•240 REM *****************                       JN
•245 BL = ASC(HB$+CHR$(0)): BL$=STR$(BL+2
 56 * ASC(LB$+CHR$(0)))                          EL
•250 POKE 214,22: POKE 211,0: PRINT RV$;Y
 E$;"  "                                         FH
•255 POKE 214,23:POKE 211,10:PRINT RV$;PU
 $;BL$;" BLOCKS FREE ";: CLOSE 6: BC=1           NE
•260 GOSUB 430: GOSUB 265:PRINT WH$ + CL$
 : END                                           CC
•265 REM ***********************                 HE
•270 REM DETERMINE USER'S CHOICE                 MG
•275 REM ***********************                 HE
•280 C=0: R=21: SC=1024: X=1864: D=54272:
 OK=0                                            KL
•285 GET BY$:IF BY$="" THEN 285                  FB
•290 IF BY$=CHR$(133) THEN GS=1: GOSUB 34
 0: GS=0: RETURN: REM DISPLAY MORE DIR.          DI
•295 IF BY$=CHR$(134) THEN RUN: REM REPEA
 T DIR.                                          MO
•300 IF BY$=CHR$(135) THEN 385: REM LOAD
 A PROGRAM                                       HM
•305 IF BY$=CHR$(136) THEN GOSUB 465: RET
 URN: REM ERASE A PROGRAM                        ED
•310 IF BY$=CHR$(17) THEN R=R+1: OK=1: IF
```

```
 R>21 THEN R=21: REM CURSOR DOWN          LC
•315 IF BY$=CHR$(145) THEN R=R-1: OK=1: I
 F R<1 THEN R=1: REM CURSOR UP            AF
•320 IF BY$=CHR$(29) THEN C=20: OK=1: REM
 CURSOR RIGHT                             PL
•325 IF BY$=CHR$(157) THEN C=0: OK=1: REM
 CURSOR LEFT                              PD
•330 IF OK=0 THEN 265                     FN
•335 GOSUB 340: GOTO 285                  HE
•340 REM ***********                      GE
•345 REM MOVE CURSOR                      KB
•350 REM ***********                      GE
•355 FOR Y=X TO X+1                       ME
•360 IF PEEK(Y) > 128 THEN POKE Y,(PEEK(Y
 )-128): POKE Y+D,1                       HE
•365 NEXT: IF GS THEN RETURN              CN
•370 POKE 211,C: POKE 214,R: PRINT UP$;:
 X=SC+(40*R)+C                            LB
•375 FOR Y=X TO X+1: IF PEEK(Y) < 128 THE
 N POKE Y,(PEEK(Y)+128): POKE Y+D,7       IB
•380 NEXT: RETURN                         EJ
•385 REM **************                   OK
•390 REM LOAD A PROGRAM                   DI
•395 REM **************                   OK
•400 CLOSE 6: Z=R:IF R>20 THEN RETURN     OP
•405 IF C=20 THEN Z=Z+20                  AG
•410 IF FI$(Z)="" THEN RETURN             BD
•415 PRINT "[CLEAR][RVSOFF]LOAD"; CHR$(34
 ); FI$(Z); CHR$(34);",8"                 FE
•420 PRINT "[4"[DOWN]"]RUN[4"[UP]"]"      EK
•425 POKE 631,19: POKE 632,13: POKE 633,1
 3: POKE 198,3: END                       LN
•430 REM ****                             DP
•435 REM BEEP                             LB
•440 REM ****                             DP
•445 POKE 54296,15: POKE 54277,7: POKE 54
 278,133                                  AK
•450 POKE 54273,28: POKE 54272,49: POKE 5
 4276,17                                  BJ
•455 FOR K=1 TO 150: NEXT: POKE 54276,18  OG
•460 POKE 54296,0: RETURN                 LH
•465 REM ***************                  CE
•470 REM ERASE A PROGRAM                  FM
•475 REM ***************                  CE
•480 POKE 214,25: POKE 211,0:PRINT RV$;"[
 4" "]ERASE -- ARE YOU SURE (Y/N)?[7" "]"
 ;                                        EL
•485 GET J$: IF J$="" THEN 485            GI
•490 IF J$ <> "Y" THEN 500                EP
•495 GOTO510                              CG
•500 GOSUB 535: GS=1:GOSUB 340: GS=0: IF
 BC THEN 250                              DK
•505 GOTO 210                             BP
•510 Z=R: IF C=20 THEN Z=Z+20             HK
•515 IF FI$(Z)="" OR R>20 THEN GOSUB535:G
 OTO 265                                  HH
•520 KL$="S0:"+FI$(Z)                     LB
•525 OPEN 15,8,15: PRINT#15, KL$: CLOSE15
 : RUN                                    PC
•535 REM ******************               LO
•540 REM PRINT COMMAND LINE               GD
•545 REM ******************               LO
•550 POKE 214,25: POKE 211,0              JI
•555 PRINT RV$+GR$+" F1=MORE  F3=REPEAT
 F5=LOAD  F7=ERASE ";: RETURN             KJ
•560 REM **********************           FC
•565 REM SET FILENAMES TO NULLS           BF
•570 REM **********************           FC
•575 FOR K=1 TO 20: FI$(K)="": NEXT: RETU
 RN                                       MI
•580 FOR K=21 TO 40: FI$(K)="": NEXT: RET
 URN                                      EM
```

SUBROUTINE (See article)

```
•5 REM  ****************                  GD
•10 REM DEFINE VARIABLES                  AC
•15 REM ****************                  GD
•20 POKE 53280,6: POKE 53281,0: POKE 646,
 1: REM SET SCREEN, TEXT COLORS           BJ
•25 COL=1                                 ND
•30 WH$=CHR$(5): GR$=CHR$(30): YE$=CHR$(1
 58): OG$=CHR$(129): PU$=CHR$(156)        LE
•35 CL$=CHR$(147): HO$=CHR$(19): RV$=CHR$
 (145)+CHR$(18): UP$=CHR$(145)            HN
•40 SP$=" ": FOR BL=0 TO 18: SP$=SP$+" "
 : NEXT                                   KB
•45 OPEN 6,8,0,"$": REM OPEN DIRECTORY    MD
•50 REM ***************************       OG
•55 REM FIND AND PRINT DISK HEADER        ND
•60 REM ***************************       OG
•65 PRINT CL$: REM CLEAR SCREEN           BE
•75 GET#6, BY$: IF BY$ <> CHR$(34) THEN 7
 5                                        IN
•80 PRINT SPC(8) "[RVSON]";OG$;BY$;       CE
•85 GET#6, BY$: IF BY$ <> "" THEN PRINT B
 Y$;: GOTO 85                             GN
•90 PRINT WH$                             FP
•95 REM  ************************         JL
•100 REM FIND AND PRINT FILENAMES         MM
•105 REM ************************         JL
•110 FOR ENTRY=0 TO 19: I=I+1: F$="": SK=
 0                                        NB
•115 GET#6, BY$,BY$:GET#6, HB$,LB$        BL
•120 GET#6, BY$: IF BY$ <> CHR$(32) THEN
 230                                      FL
•125 GET#6, BY$: IF BY$ <> CHR$(34) THEN
 125                                      IM
•130 IF COL/2=INT(COL/2) THEN PRINT TAB(1
 9)" ";                                   MJ
•135 FOR BL=1 TO 16: GET#6, BY$           CF
•140 IF BY$="" OR BY$=CHR$(34) THEN BY$=C
 HR$(32)                                  OA
•165 PRINT BY$;:NEXT                      FA
•170 GET#6, BY$                           PM
•175 GET#6, BY$: IF BY$=CHR$(32) THEN 175 MC
•180 PRINT ","; BY$; " "                  JM
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
•185 GET#6, BY$: IF BY$ <> "" THEN 185    CF
•190 NEXT: COL=COL + 1    BC
•205 IF COL=2 THEN PRINT HO$+"[DOWN]": GO
 TO 110    JC
•210 PRINT"[DOWN][12" "][RVSON]PRESS A KE
 Y[RVSOFF]    AE
•211 GETJ$:IFJ$="" THEN 211    LI
•212 PRINT HO$+"[DOWN][DOWN]";    MH
•215 IF COL/2 <> INT(COL/2) THEN FOR BL=0
  TO 19: PRINT SP$: NEXT: GOTO 225    OM
•220 FOR BL=0 TO 19: PRINT TAB(19);SP$: N
 EXT    PA
•225 PRINT SP$;SP$;HO$;"[DOWN]": GOTO 110 KC
•230 REM *****************    CJ
•235 REM PRINT BLOCKS FREE    EA
•240 REM *****************    CJ
•245 BL = ASC(HB$+CHR$(0)): BL$=STR$(BL+2
 56 * ASC(LB$+CHR$(0)))    OD
•255 POKE 214,23:POKE 211,10:PRINT RV$;PU
 $;BL$;" BLOCKS FREE ";: CLOSE 6: BC=1    JO
```

D S K D U

FROM PAGE 25

The **Bug Repellent** line codes listed to the right of the program lines of **DSKDU** are for the C-64 only. If you are typing in the program on a VIC 20, you must use the **VIC 20 Bug Repellent** line codes found on pages 106-107.

```
•1 N$="*DSKDU*":N1$="BY MARK JOHANSEN"    AB
•2 DA=8:GOTO8000    PC
•10 REM*HEX IN    DG
•12 N=0:IF LEN(H$)=0 THEN RETURN    JE
•14 FOR HX=1TOLEN(H$):N1=ASC(MID$(H$,HX,1
 )):N1=N1-48:IF N1>9 THEN N1=N1-7    PN
•16 N=N*16+N1:NEXT    FJ
•18 RETURN    IM
•1000 REM*READ    OB
•1010 PRINT"[CLEAR][RVSON]";N$:PRINT"[RVS
 ON]";N1$:PRINT    AA
•1020 INPUT"TRACK";H$:GOSUB10:T=N    ME
•1030 INPUT"SECTOR";H$:GOSUB10:S=N    CE
•1110 PRINT#15,"U1:2,0,"T;S:GOSUB9000:IF
 ER>0 THEN 1000    OF
•1120 PRINT#15,"B-P:2,0"    NP
•1130 D%(0)=0:POKE 828,PEEK(71):POKE 829,
 PEEK(72):SYS880    DG
•1160 GOSUB9000:IF ER>0 THEN 1000    BE
•2000 REM*VIEW/MOD    BL
•2010 P=0    DF
•2100 REM FLASH SCREEN    LG
•2102 IF P<0 THEN P=192    AN
•2104 IF P>192 THEN P=0    AB
•2110 PRINT"[CLEAR]TRACK ";HX$(T/16);HX$(
 T AND 15)    KH
•2120 PRINT"SECTOR ";HX$(S/16);HX$(S AND
 15)    HN
•2130 PRINT    JJ
•2140 FOR L=P TO P+63 STEP 4:PRINT HX$(L/
 16);HX$(LAND15);":";    EC
•2150 FOR BY=L TO L+3:PRINT HX$(D%(BY)/16
 );HX$(D%(BY)AND15);" ";:NEXT    DE
•2160 FOR BY=L TO L+3:C$=CHR$(D%(BY)):IF
 C$<" " OR C$>"[BACKARROW]" OR C$=CHR$(34
 ) THEN C$="."    JA
•2170 PRINT C$;:NEXT    PF
•2180 PRINT:NEXT    FA
•2200 REM*ALLOW MODS    HO
•2210 L=0:C=0    JA
•2220 BY=P+L*4+C/2:HX=CAND1:H1$(0)=HX$(D%
 (BY)/16):H1$(1)=HX$(D%(BY)AND15):C1=INT(
 C*1.5)    NC
•2225 POKE 781,L+3:POKE 782,C1+3:POKE 783
 ,0:SYS65520    LL
•2230 POKE 204,0    JJ
•2240 GET I$:IF I$="" THEN 2240    KO
•2250 POKE 204,1:PRINT H1$(HX);    CB
•2260 IF I$="[F1]" THEN P=P-64:GOTO2100    OA
•2270 IF I$="[F3]" THEN P=P+64:GOTO2100    EK
•2280 IF I$="[F7]" THEN 1000    NF
•2285 IF I$="[F4]" THEN T=D%(0):S=D%(1):G
 OTO1110    KP
•2290 IF I$="[DOWN]" THEN 2750    GA
•2300 IF I$="[RIGHT]" THEN 2700    HL
•2310 IF I$="[UP]" THEN 2800    BJ
•2320 IF I$="[LEFT]" THEN 2850    PG
•2330 IF I$="[F8]" THEN 3000    OD
•2340 IF I$="[F2]" THEN 4000    OD
•2350 IF I$="[HOME]" THEN 2210    IF
•2400 IF I$<"0" OR I$>"F" OR I$>"9" AND I
 $<"A" THEN 7000    OA
•2410 PRINT "[LEFT]"I$;    CJ
•2430 H1$(HX)=I$:H$=H1$(0)+H1$(1)    AB
•2450 GOSUB10:D%(BY)=N    IO
•2460 POKE 781,L+3:POKE 782,C/2+15:POKE 7
 83,0:SYS65520    BJ
•2470 C$=CHR$(D%(BY)):IF C$<" " OR C$>"[B
 ACKARROW]" OR C$=CHR$(34) THEN C$="."    II
•2480 PRINT C$;    DM
•2490 REM FALL THRU INTO CURSOR RIGHT    LO
•2700 REM CURSOR RIGHT    NK
•2710 C=C+1:IF C>7 THEN C=0:GOTO2750    HJ
•2720 GOTO2220    EP
•2750 REM CURSOR DOWN    LM
•2760 L=L+1:IF L>15 THEN L=0    FP
•2770 GOTO2220    EP
•2800 REM CURSOR UP    CK
•2810 L=L-1:IF L<0 THEN L=15    CA
•2820 GOTO2220    EP
•2850 REM CURSOR LEFT    LJ
•2860 C=C-1:IF C<0 THEN C=7:GOTO2800    KO
•2870 GOTO2220    EP
•3000 REM*WRITE    FL
```

```
·3010 PRINT"[CLEAR]PREPARING TO WRITE"    NG
·3020 PRINT" TRACK ";HX$(T/16);HX$(T AND1
 5);" SECTOR ";HX$(S/16);HX$(S AND15)     CJ
·3030 PRINT:PRINT"<RET> TO WRITE":PRINT"A
 NY OTHER KEY TO ABORT"                   BM
·3040 GET I$:IF I$="" THEN 3040           NB
·3050 IF I$<>CHR$(13) THEN 2100           CJ
·3100 REM DO WRITE                        LE
·3110 PRINT#15,"B-P:2,0"                  NP
·3120 PRINT"WRITING [3"."]"               LI
·3130 D%(0)=D%(0):POKE 828,PEEK(71):POKE
 829,PEEK(72):SYS910                      GE
·3140 PRINT#15,"U2:2,0,";T;S:GOSUB9000:IF
 ER THEN 2100                             AP
·3150 GOTO2100                            EI
·4000 REM*QUIT                            AG
·4010 CLOSE2:CLOSE15                      BI
·4020 PRINT"[CLEAR]DONE [3"."]"           FO
·4030 POKE 204,0:END                      FO
·7000 REM*HELP                            OO
·7010 PRINT"[CLEAR]DSKDU HELP":PRINT      PN
·7020 PRINT"F1 PREVIOUS PAGE"             BL
·7030 PRINT"F3 NEXT PAGE"                 PJ
·7050 PRINT"F7 READ SECTOR"               HC
·7060 PRINT"F4 READ NEXT SECTOR"          NL
·7065 PRINT"[3" "]IN FILE"                PA
·7070 PRINT"F8 WRITE SECTOR"              BA
·7080 PRINT"CRSR CONTROLS"                KF
·7090 PRINT"[3" "]UP,DOWN,LEFT,RIGHT"     IP
·7100 PRINT"HOME TOP LEFT CORNER"         KN
·7120 PRINT"0-9,A-F"                      GN
·7130 PRINT"[3" "]OVERWRITE NIBBLE"       IC
·7140 PRINT"F2 QUIT"                      MD
·7200 PRINT:PRINT"(ANY KEY TO CONT)"      FC
·7210 GET I$:IF I$="" THEN 7210           MO
·7220 GOTO2100                            EI
·8000 REM*INIT                            AD
·8020 OPEN15,DA,15                        GA
·8030 OPEN2,DA,2,"#"                      IM
·8040 DIM D%(255)                         ML
·8050 DIM HX$(15):FOR BY=0TO15:READ HX$(B
 Y):NEXT                                  FP
·8060 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,
 F                                        BK
·8070 DIM H1$(1)                          KK
·8100 REM LOAD ML ROUTINES                PO
·8110 READ AD:IF AD<0 THEN 8190           LA
·8120 READ B:IF B>=0 THEN POKE AD,B:AD=AD
 +1:GOTO 8120                             BK
·8130 GOTO 8110                           FJ
·8190 GOTO1000                            FC
·8200 REM READ/WRITE ROUTINES             BL
·8210 DATA 840:REM SETUP                  IO
·8220 DATA 173,60,3:REM LDA 828           HL
·8230 DATA 174,61,3:REM LDX 829           MJ
·8240 DATA 133,251:REM STA !251           GO
·8250 DATA 134,252:REM STX !252           HA
·8260 DATA 169,0:REM LDA #0               IG
·8270 DATA 141,62,3:REM STA 830           GE
·8280 DATA 96,-1:REM RTS                  DD
·8300 DATA 860:REM ADVANCE POINTER        NC
·8310 DATA 24:REM CLC                     JD
·8320 DATA 165,251:REM LDA !251           EE
·8330 DATA 105,2:REM ADC #2               HB
·8340 DATA 133,251:REM STA !251           GO
·8350 DATA 165,252:REM LDA !252           CK
·8360 DATA 105,0:REM ADC #0               GN
·8370 DATA 133,252:REM STA !252           DM
·8380 DATA 96,-1:REM RTS                  DD
·8400 DATA 880:REM READ ROUTINE           EI
·8410 DATA 162,2:REM LDX #2               KA
·8420 DATA 32,198,255:REM JSR CHKIN       MP
·8430 DATA 32,72,3:REM JSR SETUP          DB
·8440 DATA 32,207,255:REM JSR CHRIN       OH
·8450 DATA 160,1:REM LDY #1               IH
·8460 DATA 145,251:REM STA (251,Y         BE
·8470 DATA 32,92,3:REM JSR ADVPTR         MM
·8480 DATA 206,62,3:REM DEC 830           GM
·8490 DATA 208,241:REM BNE -15            AG
·8500 DATA 32,204,255:REM JSR CLRCHN      DI
·8510 DATA 96,-1:REM RTS                  DD
·8600 DATA 910:REM WRITE ROUTINE          HF
·8610 DATA 162,2:REM LDX #2               KA
·8620 DATA 32,201,255:REM JSR CHKOUT      MB
·8630 DATA 32,72,3:REM JSR SETUP          DB
·8640 DATA 160,1:REM LDY #1               IH
·8650 DATA 177,251:REM LDA (251,Y         BG
·8660 DATA 32,210,255:REM JSR CHROUT      OK
·8670 DATA 32,92,3:REM JSR ADVPTR         MM
·8680 DATA 206,62,3:REM DEC 830           GM
·8690 DATA 208,241:REM BNE -15            AG
·8700 DATA 32,204,255:REM JSR CLRCHN      DI
·8710 DATA 96,-1:REM RTS                  DD
·8720 DATA -1                             OE
·9000 REM DISK CHECK                      DG
·9020 INPUT#15,ER,ER$,ET,ES               EC
·9030 IF ER=0 THEN RETURN                 EM
·9050 PRINT "[CLEAR][RVSON]";ER;ER$:PRINT
 "[RVSON]";ET;ES                          EF
·9060 PRINT:PRINT"F2 TO QUIT":PRINT"ANY O
 THER KEY TO CONT"                        HM
·9070 GET I$:IF I$="" THEN 9070           NM
·9080 IF I$="[F2]" OR I$="[F1]" THEN 4000 GH
·9090 RETURN                              IM
```

VIC 20 BUG REPELLENT LINE CODES FOR DSKDU

```
# 1  :AB  # 12 :JE  # 18   :IM  # 1020:ME  # 1120:NP  # 2000:BL  # 2102:AN  # 2120:HN
# 2  :PC  # 14 :PN  # 1000:OB  # 1030:CE  # 1130:DG  # 2010:DF  # 2104:AB  # 2130:JJ
# 10 :DG  # 16 :FJ  # 1010:AA  # 1110:OF  # 1160:BE  # 2100:LG  # 2110:KH  # 2140:EC
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering ***Ahoy!*** programs. Refer to these pages **before** entering any programs!

# 2150:DE	# 2320:PG	# 2800:CK	# 4000:AG	# 7140:MD	# 8200:BL	# 8380:DD	# 8640:IH
# 2160:JA	# 2330:OD	# 2810:CA	# 4010:BI	# 7200:FC	# 8210:IO	# 8400:EI	# 8650:BG
# 2170:PF	# 2340:OD	# 2820:EP	# 4020:FO	# 7210:MO	# 8220:HL	# 8410:KA	# 8660:OK
# 2180:FA	# 2350:IF	# 2850:LJ	# 4030:FO	# 7220:EI	# 8230:MJ	# 8420:MP	# 8670:MM
# 2200:HO	# 2400:OA	# 2860:KO	# 7000:OO	# 8000:BI	# 8240:GO	# 8430:DB	# 8680:GM
# 2210:JA	# 2410:CJ	# 2870:EP	# 7010:PN	# 8001:AD	# 8250:HA	# 8440:OH	# 8690:AG
# 2220:NC	# 2430:AB	# 3000:FL	# 7020:BL	# 8020:GA	# 8260:IG	# 8450:IH	# 8700:DI
# 2225:LL	# 2450:IO	# 3010:NG	# 7030:PJ	# 8030:IM	# 8270:GE	# 8460:BE	# 8710:DD
# 2230:JJ	# 2460:BJ	# 3020:CJ	# 7050:HC	# 8040:ML	# 8280:DD	# 8470:MM	# 8720:OE
# 2240:KO	# 2470:II	# 3030:BM	# 7060:NL	# 8050:FP	# 8300:NC	# 8480:GM	# 9000:DG
# 2250:CB	# 2480:DM	# 3040:NB	# 7065:PA	# 8060:BK	# 8310:JD	# 8490:AG	# 9020:EC
# 2260:OA	# 2490:LO	# 3050:CJ	# 7070:BA	# 8070:KK	# 8320:EE	# 8500:DI	# 9030:EM
# 2270:EK	# 2700:NK	# 3100:LE	# 7080:KF	# 8100:PO	# 8330:HB	# 8510:DD	# 9050:EF
# 2280:NF	# 2710:HJ	# 3110:NP	# 7090:IP	# 8110:LA	# 8340:GO	# 8600:HF	# 9060:HM
# 2285:KP	# 2720:EP	# 3120:LI	# 7100:KN	# 8120:BK	# 8350:CK	# 8610:KA	# 9070:NM
# 2290:GA	# 2750:LM	# 3130:GE	# 7120:GN	# 8130:FJ	# 8360:GN	# 8620:MB	# 9080:GH
# 2300:HL	# 2760:FP	# 3140:AP	# 7130:IC	# 8190:FC	# 8370:DM	# 8630:DB	# 9090:IM
# 2310:BJ	# 2770:EP	# 3150:EI					

THE HARDWARE INTERRUPT WEDGE

FROM PAGE 52

VIC 20 VERSION

```
•10 REM***          VIC 20          ***      HL
•12 REM***SAMPLE INTERRUPT ROUTINE***        LD
•15 REM***      BASIC LOADER        ***      BJ
•20 REM***LOADS INTO CASSETTE BUFF***        IG
•25 REM***  'SYS828' TO ACTIVATE   ***       HK
•30 REM*** 'SYS854' TO DEACTIVATE ***        FH
•40 FORX=828TO866                            KF
•50 READA:POKEX,A:NEXT:END                   OH
•100 DATA120,169,73,141,20,3,169,3,141,21
 ,3,88,96,165,197,141,255,30,169,4           IP
•200 DATA141,255,150,76,191,234,120,169,1
 91,141,20,3,169,234,141,21,3,88,96          AO
```

C-64 VERSION

```
•10 REM***          C64             ***      CA
•12 REM***SAMPLE INTERRUPT ROUTINE***        GI
•15 REM***      BASIC LOADER        ***      KM
•20 REM***LOADS INTO CASSETTE BUFF***        DM
•25 REM***  'SYS828' TO ACTIVATE   ***       CE
•30 REM*** 'SYS854' TO DEACTIVATE ***        BN
•40 FORX=828TO866                            KF
•50 READA:POKEX,A:NEXT:END                   OH
•100 DATA120,169,73,141,20,3,169,3,141,21
 ,3,88,96,165,197,141,255,4,169,4            BE
•200 DATA141,255,216,76,49,234,120,169,49
 ,141,20,3,169,234,141,21,3,88,96            AM
```

SUPER DUPER

FROM PAGE 29

```
•5 POKE53280,6:POKE53281,0:POKE646,1         CM
•10 PRINTCHR$(147)                           FG
•20 FORX=3584TO3660:READZ:POKEX,Z:NEXTX      AF
•30 POKE51,0:POKE52,15:POKE251,0:POKE252,
 15:POKE828,0                                PH
•40 PRINT"SELECT OPTION"                     JG
•50 PRINT:PRINT"1. COPY SEQ FILE"            KO
•60 PRINT:PRINT"2. COPY PGM FILE"            MC
•70 PRINT:PRINT"3. END"                      BC
•80 PRINT                                    JJ
•90 GETR$:IFR$=""THEN90                      EA
•100 A=VAL(R$):IFA<1ORA>3THEN90              EH
•110 ONAGOTO120,140,290                      IK
•120 INPUT"SEQ FILENAME";AN$                 OL
•130 F$="S":GOTO160                          BA
•140 INPUT"PGM FILENAME";AN$                 OA
•150 F$="P"                                  LE
•160 OPEN3,8,3,AN$                           JA
•170 SYS3584:CLOSE3                          HG
•180 IFPEEK(252)=159THENPRINT"FILE TOO LO
 NG":STOP                                    NK
•190 POKE253,PEEK(828):POKE254,PEEK(252):
 POKE251,0:POKE252,15:POKE828,0              IO
•200 PRINT"SWAP DISKS IF PUTTING COPY ON
 A NEW DISK":PRINT                           NB
•210 INPUT"NAME OF NEW FILE";AN$             LB
•220 OPEN3,8,3,AN$+","+F$+",W":SYS3618:CL
 OSE3                                        CN
•230 PRINT"- COPY COMPLETE -":GOTO30         CB
•240 DATA 162,3,32,198,255,32,228,255,172
 ,60,3,145,251,200,140,60,3,192              DA
•250 DATA 0,208,8,230,252,166,252,224,159
 ,240,4,165,144,240,228,32,204,255           LI
•260 DATA 96,162,3,32,201,255,172,60,3,17
 7,251,196,253,208,6,166,252                 MD
•270 DATA 228,254,240,16,200,140,60,3,32,
 210,255,192,0,208,230,230,252,76            CM
•280 DATA 42,14,32,204,255,96                HL
•290 END                                     IC
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

COMMODORE FONT EDITOR

FROM PAGE 82

```
•10 REM COMMODORE FONT EDITOR                    FH
•20 REM PROGRAMMED BY STEPHEN REED               EC
•30 REM                                          JD
•40 REM IF YOU DON'T WANT TO TYPE IN             JO
•50 REM THIS PROGRAM, COPIES ARE                 NG
•60 REM AVAILABLE ON DISK WITH SEVERAL           KH
•70 REM CHARACTER FONTS. SEND $12.95 TO:         FK
•80 REM                                          JD
•90 REM        FONT EDITOR                       LG
•100 REM       2404 S. MYRTLE AVE.               LJ
•110 REM       SANFORD, FL  32707                IE
•120 REM                                         JD
•130 REM                                         JD
•140 REM ****************************            CE
•150 REM * INITIALIZE PROGRAM DATA *             DE
•160 REM ****************************            CE
•170 REM                                         JD
•180 POKE53281,1:POKE53280,7                     BP
•190 POKE52,48:POKE56,48:CLR:DIMR$(7)            PC
•200 FORA=0TO7:READR(A):NEXTA                    AA
•210 DATA 128,64,32,16,8,4,2,1                   BJ
•220 PRINT"[CLEAR]"CHR$(142)CHR$(8)"[BLUE
][RVSON]";                                       CM
•230 CC=1:OC=1                                   CI
•240 PRINT"[9" "]COMMODORE FONT EDITOR[9"
"][SS]";                                         EC
•250 PRINT"[BLACK][RVSON][40"[c T]"]"            BM
•260 GOSUB510:REM GET E WINDOW                   AG
•270 GOSUB740:REM MENU ON SCREEN                 HD
•280 GOSUB920:REM DISPLAY CHARACTERS             CC
•290 GOSUB1070:REM UPDATE SCRN FOR CC            GK
•300 IFPEEK(12288)=60THEN430                     AG
•310 REM                                         JD
•320 REM ****************************            CE
•330 REM * DOWNLOAD ROM CHARACTERS *             FA
•340 REM ****************************            CE
•350 REM                                         JD
•360 PRINT"[HOME][16"[DOWN]"]";O$;"[BLUE]
PLEASE WAIT 30 SECONDS"                          HM
•370 POKE56334,PEEK(56334)AND254                 DN
•380 POKE1,PEEK(1)AND251                         IM
•390 FORA=0TO2047                                CP
•400 POKEA+12288,PEEK(A+53248):NEXT              JD
•410 POKE1,PEEK(1)OR4                            BE
•420 POKE56334,PEEK(56334)OR1:GOSUB1070          IB
•430 GOSUB740                                    CL
•440 POKE53272,(PEEK(53272)AND240)+12            LI
•450 GOSUB1390:IFA$<"0"ORA$>"9"THEN450           DJ
•460 A=VAL(A$):IFA=0THENA=10                     IB
•470 ONAGOSUB1700,1890,2070,2430,2620,275
0,2920,3010,3170,3240                            LN
•480 GOTO 450                                    CN
•490 REM                                         JD
•500 REM ********************************        NK
•510 REM * PUT EDIT WINDOW ON SCREEN  *          DC
•520 REM ********************************        NK
•530 REM                                         JD
•540 PRINT"[HOME][3"[DOWN]"] [BLACK][RVSO
N]";                                             BJ
•550 PRINT"[3" "]12345678[3" "]"                 CJ
•560 PRINT" [RVSON] [PURPLE][s M][10" "][
s N][BLACK] "                                    GO
•570 PRINT" [RVSON] [PURPLE] [s M][8" "][
s N] [BLACK] "                                   EM
•580 PRINT" [RVSON]1[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]1"                             ID
•590 PRINT" [RVSON]2[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]2"                             KH
•600 PRINT" [RVSON]3[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]3"                             OH
•610 PRINT" [RVSON]4[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]4"                             PL
•620 PRINT" [RVSON]5[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]5"                             PD
•630 PRINT" [RVSON]6[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]6"                             NH
•640 PRINT" [RVSON]7[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]7"                             OH
•650 PRINT" [RVSON]8[PURPLE]  [RVSOFF][8"
"][RVSON]  [BLACK]8"                             OL
•660 PRINT" [RVSON] [PURPLE] [s N][8" "][
s M] [BLACK] "                                   EM
•670 PRINT" [RVSON] [PURPLE][s N][10" "][
s M][BLACK] "                                    HC
•680 PRINT" [RVSON][3" "]12345678[3" "][R
VSOFF]":RETURN                                   OF
•690 REM                                         JD
•700 REM *****************************           CE
•710 REM * PUT MAIN MENU ON SCREEM *             HG
•720 REM *****************************           CE
•730 REM                                         JD
•740 PRINT"[HOME][3"[DOWN]"][22"[RIGHT]"]
[RED][RVSON][3"*"] MENU [3"*"][BLACK]"           DN
•750 O$="[17"[RIGHT]"]"                          GJ
•760 PRINT:PRINTO$;"1] SAVE FONT"                MB
•770 PRINTO$;"2] LOAD FONT"                      LB
•780 PRINTO$;"3] EDIT CHARACTER"                 DB
•790 PRINTO$;"4] MOVE CHARACTERS"                GK
•800 PRINTO$;"5] SWAP CHARACTERS"                LF
•810 PRINTO$;"6] INVERSE CHARACTERS"             JN
•820 PRINTO$;"7] GET ROM CHARACTERS"             EL
•830 PRINTO$;"8] DISPLAY DATA VALUES"            EE
•840 PRINTO$;"9] DECREMENT DISPLAY"              CE
•850 PRINTO$;"0] INCREMENT DISPLAY"              FJ
•860 PRINT:PRINTO$;"[GREEN][3" "]SELECT O
PTION?[5" "]":RETURN                             KM
•870 REM                                         JD
•880 REM ********************************        NK
•890 REM * PUT CHARACTER DATA ON LINE *          BC
```

```
•900 REM ******************************  NK
•910 REM                                 JD
•920 PRINT"[HOME][18"[DOWN]"]";          NP
•930 PRINT"[24" "]";                     BJ
•940 PRINT"[BLACK]@ABCDEFGHIJKLMNO";     FC
•950 PRINT"PQRSTUVWXYZ[[EP]][UPARROW][BAC
 KARROW] !"+CHR$(34)+"#$%&'()*+,-./012345
 67";                                    KM
•960 PRINT"89:;<=>?[s *][s A][s B][s C][s
 D][s E][s F][s G][s H][s I][s J][s K][s
 L][s M][s N][s O][s P][s Q][s R][s S][s
 T][s U][s V][s W][s X][s Y][s Z][s +][c
 -][s -][PI][c *]"                       HB
•970 FORA=94TO255                        DD
•980 POKE1024+18*40+24+A,A               NP
•990 POKE55296+18*40+24+A,0              BL
•1000 NEXT                               IA
•1010 PRINTCHR$(146):RETURN              CC
•1020 REM                                JD
•1030 REM ****************************** NK
•1040 REM * PUT CURRENT DATA ON SCREEN * EO
•1050 REM ****************************** NK
•1060 REM                                JD
•1070 POKE55296+18*40+OC+24,0:OC=CC      HH
•1080 POKE55296+18*40+CC+24,2            HC
•1090 PRINT"[HOME][15"[DOWN]"][5"[RIGHT]"
 ][PURPLE][RVSON]CODE ";RIGHT$(STR$(CC),L
 EN(STR$(CC))-1);                        AD
•1100 IFCC<10THENPRINT"  "               OE
•1110 IFCC<100THENPRINT" "               BH
•1120 PRINT"[HOME][6"[DOWN]"][RVSOFF]";:F
 ORA=0TO7:P$=""                          FL
•1130 B=PEEK(12288+A+(CC*8))             CJ
•1140 IFB=>128THENZP=1:B=B-128           PC
•1150 GOSUB1320                          FH
•1160 IFB=>64THENZP=1:B=B-64             IH
•1170 GOSUB1320                          FH
•1180 IFB=>32THENZP=1:B=B-32             DJ
•1190 GOSUB1320                          FH
•1200 IFB=>16THENZP=1:B=B-16             DN
•1210 GOSUB1320                          FH
•1220 IFB=>8THENZP=1:B=B-8               AE
•1230 GOSUB1320                          FH
•1240 IFB=>4THENZP=1:B=B-4               NE
•1250 GOSUB1320                          FH
•1260 IFB=>2THENZP=1:B=B-2               PA
•1270 GOSUB1320                          FH
•1280 IFB=1THENZP=1                      OK
•1290 GOSUB1320                          FH
•1300 PRINT"[BLACK][4"[RIGHT]"]";P$:NEXTA HL
•1310 RETURN                             IM
•1320 IFZP=1THENP$=P$+"[s Q]":ZP=0:RETURN MK
•1330 P$=P$+" ":RETURN                   HC
•1340 REM                                JD
•1350 REM **************************     CE
•1360 REM * SCAN KEYBOARD FOR INPUT *    KP
•1370 REM **************************     CE
•1380 REM                                JD
```

```
•1390 GETA$:IFA$=""THEN1390              KB
•1400 RETURN                             IM
•1410 REM                                JD
•1420 REM **************************     CE
•1430 REM *     ERASE MENU AREA     *    ME
•1440 REM **************************     CE
•1450 REM                                JD
•1460 PRINT"[HOME][3"[DOWN]"]";          GL
•1470 FORA=0TO13                         MN
•1480 PRINTO$;"[22" "]"                  FG
•1490 NEXTA:RETURN                       JA
•1500 REM                                JD
•1510 REM **************************     CE
•1520 REM * FORMATTED NUMERIC INPUT *    FJ
•1530 REM **************************     CE
•1540 REM                                JD
•1550 PRINT"[3"[c D]"][3"[LEFT]"]";:Z=0:B
 $=""                                    FA
•1560 FORA=1TO3:GOSUB1600:IFZ=1THEN1580  HH
•1570 NEXT                               IA
•1580 W=VAL(B$):IFW>255THENPRINT"[3"[LEFT
 ]"]";:GOTO1550                          PF
•1590 RETURN                             IM
•1600 GOSUB1390                          FI
•1610 IFA$=CHR$(13)THENZ=1:RETURN        NH
•1620 IFA$<"0"ORA$>"9"THEN1600           ON
•1630 B$=B$+A$                           IG
•1640 PRINTA$;:RETURN                    BH
•1650 REM                                JD
•1660 REM *************************      OG
•1670 REM * SAVE FONT DATA TO DISK *     FH
•1680 REM *************************      OG
•1690 REM                                JD
•1700 GOSUB1430                          FP
•1710 PRINT"[HOME][4"[DOWN]"]"O$"  [GREEN
 ][3"."]SAVE DATA[3"."][BLACK]"          NC
•1720 PRINT:PRINT"[10"[DOWN]"]"O$"FILESPE
 C";                                     OM
•1730 INPUTN$:IFLEN(N$)>8THEN1700        JO
•1740 OPEN 1,8,12,N$+",P,W"              FA
•1750 PRINT#1,CHR$(0)+CHR$(48);          NA
•1760 FORI=0TO255                        AB
•1770 PRINT"[HOME][8"[DOWN]"]"O$"[8"[RIGH
 T]"]";I                                 KI
•1780 B$=""                              FO
•1790 FORJ=0TO7                          IC
•1800 B$=B$+CHR$(PEEK(12288+J+I*8)):NEXT MF
•1810 PRINT#1,B$;                        KC
•1820 NEXTI:CLOSE1:GOSUB1430             GG
•1830 GOSUB740:RETURN                    PK
•1840 REM                                JD
•1850 REM ****************************   GH
•1860 REM * LOAD FONT DATA FROM DISK *   EE
•1870 REM ****************************   GH
•1880 REM                                JD
•1890 GOSUB1430                          FP
•1900 PRINT"[HOME][4"[DOWN]"]"O$"  [GREEN
 ][3"."]LOAD DATA[3"."][BLACK]"          DF
```

```
•1910 PRINT:PRINT"[10"[DOWN]"]"O$"FILESPE
 C";                                      OM
•1920 INPUTN$:IFLEN(N$)>8THEN1700         JO
•1930 OPEN 1,8,12,N$+",P,R"               CN
•1940 GET#1,A$,A$                         AL
•1950 FORI=0TO255                         AB
•1960 PRINT"[HOME][8"[DOWN]"]"O$"[8"[RIGH
 T]"]";I                                  KI
•1970 FORJ=0TO7                           IC
•1980 GET#1,A$:A=0:IFA$<>""THENA=ASC(A$)  KJ
•1990 POKE12288+J+I*8,A:NEXTJ             BO
•2000 NEXTI:CLOSE1:GOSUB1430              GG
•2010 GOSUB740:RETURN                     PK
•2020 REM                                 JD
•2030 REM ***************************     OG
•2040 REM * EDIT CHARACTER ROUTINE *      PL
•2050 REM ***************************     OG
•2060 REM                                 JD
•2070 GOSUB1430                           FP
•2080 PRINT"[HOME][4"[DOWN]"]"O$"[PURPLE]
   EDIT CHARACTER[BLACK]"                 OM
•2090 PRINT:PRINTO$"CHARACTER # ";:GOSUB1
 550                                      MM
•2100 CC=W:GOSUB1070                      CM
•2110 GOSUB1430:PRINT"[HOME][4"[DOWN]"]"O
 $;"[PURPLE]EDIT CHARACTER[BLACK]"        NN
•2120 PRINT:PRINTO$"[E][6" "]ERASE BUFFER
 "                                        GO
•2130 PRINTO$"[*]/[.]  SET PIXEL"         FN
•2140 PRINTO$"[SPACE]  RESET PIXEL":PRINT
 O$"[RETURN] FINISHED"                    KE
•2150 PRINT:PRINT:PRINTO$" CURSOR KEYS MO
 VE"                                      BK
•2160 PRINTO$" EDITING CURSOR."           NI
•2170 X=4:Y=4:G=1:GOSUB2350               KC
•2180 GOSUB1390:IFA$="E"THENGOSUB2410:GOT
 O2180                                    GE
•2190 IFA$=" "THENG=3:GOSUB2350:GOTO2180  OA
•2200 IFA$="*"ORA$="."THENG=2:GOSUB2350:G
 OTO2180                                  CE
•2210 IFA$=CHR$(13)THEN2280               IP
•2220 IFA$<>"[UP]"ANDA$<>"[DOWN]"ANDA$<>"
 [LEFT]"ANDA$<>"[RIGHT]"THEN2180          PI
•2230 IFA$="[UP]"ANDY>1THENG=0:GOSUB2400:
 Y=Y-1:GOSUB2350:GOTO2180                 DM
•2240 IFA$="[DOWN]"ANDY<8THENG=0:GOSUB240
 0:Y=Y+1:GOSUB2350:GOTO2180               GF
•2250 IFA$="[RIGHT]"ANDX<8THENG=0:GOSUB24
 00:X=X+1:GOSUB2350:GOTO2180              BM
•2260 IFA$="[LEFT]"ANDX>1THENG=0:GOSUB240
 0:X=X-1:GOSUB2350:GOTO2180               OH
•2270 GOTO 2180                           FA
•2280 FORA=0TO7                           IP
•2290 GOSUB2320:REM GET BYTE IN B         OA
•2300 POKE12288+(CC*8)+A,B:NEXTA          II
•2310 GOSUB1070:GOSUB1430:GOSUB740:RETURN PO
•2320 B=0:FORR=0TO7                       AA
•2330 U=PEEK(1268+(A*40)+R):IFU=87ORU=81T
 HENB=B+R(R)                              GG
•2340 NEXTR:RETURN                        KB
•2350 X$="[8"[RIGHT]"]":Y$="[8"[DOWN]"]":
 PRINT"[HOME][5"[DOWN]"][3"[RIGHT]"]";LEF
 T$(X$,X);LEFT$(Y$,Y);                    LJ
•2360 IFG=2THENPRINT"[BLACK][s Q]":G=1:GO
 SUB2350:RETURN                           KC
•2370 IFG=3THENPRINT"[BLACK] ":G=1:GOSUB2
 350:RETURN                               BB
•2380 K=PEEK(1227+X+Y*40):IFK=81THENPRINT
 "[RED][s W]":RETURN                      GC
•2390 PRINT"[RED]+":RETURN                OI
•2400 POKE1227+X+Y*40,K:POKE55499+X+Y*40,
 0:RETURN                                 LG
•2410 PRINT"[HOME][6"[DOWN]"]";:FORA=1TO8
 :PRINT"[4"[RIGHT]"][8" "]"               AM
•2420 NEXTA:G=1:K=32:GOSUB2350:RETURN     DH
•2430 GOSUB1430                           FP
•2440 PRINT"[HOME][4"[DOWN]"]"O$"STARTING
  CHAR ";                                 NA
•2450 GOSUB1550:M1=W                      BJ
•2460 PRINT:PRINTO$"ENDING CHAR ";:GOSUB1
 550                                      EH
•2470 M2=W                                IJ
•2480 PRINT:PRINTO$"MOVE TO CHAR ";:GOSUB
 1550                                     MO
•2490 IFM1>M2THEN2430                     LM
•2500 IFW<=M2 ANDW=>M1 THEN 2430          BP
•2510 PRINT:PRINTO$;"[BLUE]  [3"."]MOVING
 [3"."][BLACK]"                           FP
•2520 FORA=0TO(M2-M1)                     EM
•2530 FORB=0TO7                           HK
•2540 POKE12288+B+(A+W)*8,PEEK(12288+B+(A
 +M1)*8)                                  DE
•2550 NEXTB,A                             DI
•2560 GOSUB1430:GOSUB740:RETURN           IK
•2570 REM                                 JD
•2580 REM *****************************   GH
•2590 REM * SWAP CHARACTER DATA SUBR *    DK
•2600 REM *****************************   GH
•2610 REM                                 JD
•2620 GOSUB1430:PRINT"[BLACK][HOME][4"[DO
 WN]"]"O$"STARTING CHAR ";:GOSUB1550:SC=W HF
•2630 PRINT:PRINTO$"ENDING CHAR ";:GOSUB1
 550:EC=W                                 AL
•2640 PRINT:PRINTO$"SWAP WITH ";:GOSUB155
 0                                        BA
•2650 FORA=0TO(EC-SC):FORB=0TO7           CL
•2660 T=PEEK(12288+B+(A+W)*8)             BM
•2670 POKE12288+B+(A+W)*8,PEEK(12288+B+(A
 +SC)*8)                                  MA
•2680 POKE12288+B+(A+SC)*8,T:NEXTB,A      JK
•2690 GOSUB1430:GOSUB740:GOSUB1070:RETURN EI
•2700 REM                                 JD
•2710 REM ***************************     OG
•2720 REM * REVERSE CHARACTER DATA *      KH
•2730 REM ***************************     OG
•2740 REM                                 JD
```

IMPORTANT! Letters on white background are **Bug Repellent** line codes. **Do not enter them!** Pages 85 and 86 explain these codes and provide other essential information on entering *Ahoy!* programs. Refer to these pages **before** entering any programs!

```
•2750 GOSUB1430                                              FP
•2760 PRINT"[HOME][4"[DOWN]"]";:IFYT=0THENPRINTO$"[BLACK]REVERSE DATA"   BA
•2770 PRINT:PRINTO$"STARTING CHAR ";:GOSUB1550               AF
•2780 SC=W                                                   KE
•2790 PRINT:PRINT:PRINTO$"ENDING CHAR ";:GOSUB1550           LF
•2800 IFW<SCTHEN2750                                         LK
•2810 IFYT=1THENRETURN                                       FP
•2820 PRINT:PRINT:PRINTO$"  [BLUE][3"."]REVERSING[3"."][BLACK]"   OM
•2830 FORA=0TO(W-SC):FORB=0TO7                               OB
•2840 POKE12288+B+(A+SC)*8,255-PEEK(12288+B+(A+SC)*8):NEXTB,A   DG
•2850 GOSUB1430:GOSUB740                                     LH
•2860 GOSUB1070:RETURN                                       CO
•2870 REM                                                    JD
•2880 REM **************************                         OG
•2890 REM * GET ROM CHARACTER DATA *                         FC
•2900 REM **************************                         OG
•2910 REM                                                    JD
•2920 YT=1:PRINT"[BLACK]":GOSUB2750:YT=0                     IG
•2930 PRINT:PRINTO$"[BLUE][3"."]LOADING FROM ROM[3"."][BLACK]"   DK
•2940 POKE56334,PEEK(56334)AND254                            DN
•2950 POKE1,PEEK(1)AND251                                    IM
•2960 FORA=SC*8TOSC*8+(W-SC)*8+7                             IL
•2970 POKEA+12288,PEEK(A+53248):NEXT                         JD
•2980 POKE1,PEEK(1)OR4                                       BE
•2990 POKE56334,PEEK(56334)OR1                               PD
•3000 GOSUB1430:GOSUB740:GOSUB1070:RETURN                    EI
•3010 GOSUB1430                                              FP
•3020 PRINT"[HOME][4"[DOWN]"]"O$;"[BLACK]CHARACTER NUMBER ";   IL
•3030 GOSUB1550                                              GG
•3040 PRINT:PRINT:TP=CC:CC=W:GOSUB1070                       JP
•3050 PRINT"[HOME][5"[DOWN]"]":FORA=0TO7                     PL
•3060 PRINTO$;PEEK(12288+A+(W*8))                            CP
•3070 NEXTA                                                  MF
•3080 PRINT:PRINTO$;"[4" "]SPACE TO RETURN"                  HK
•3090 GOSUB1390:CC=TP:GOSUB1070                              EL
•3100 GOSUB1430:GOSUB740                                     LH
•3110 GOSUB1070:RETURN                                       CO
•3120 REM                                                    JD
•3130 REM ***************************                        CE
•3140 REM * DECREMENT CHARACTER --> *                        JG
•3150 REM ***************************                        CE
•3160 REM                                                    JD
•3170 CC=CC-1:IFCC<0THENCC=255                               NC
•3180 GOSUB1070:RETURN                                       CO
•3190 REM                                                    JD
•3200 REM ***************************                        CE
•3210 REM * INCREMENT CHARACTER --> *                        IK
•3220 REM ***************************                        CE
•3230 REM                                                    JD
•3240 CC=CC+1:IFCC>255THENCC=1                               BG
•3250 GOTO 3180                                              FB
```

TILE TIME

FROM PAGE 81

```
•1 REM --------- THE TILE GAME ----------                    MK
•2 REM   (C) MURDOCK & BLACKBURN '84                         FG
•3 REM          10/15/1984                                   HL
•4 REM                                                       JD
•100 FORX=1TO6:READCH:CO$(X)=CHR$(CH):NEXT                   MG
•110 DATA 28,158,156,30,150,5                                JE
•120 B1$=CHR$(154):B2$=CHR$(31):B3$=CHR$(152):C1$=B1$:C2$=CHR$(144)   LB
•130 SC$=CHR$(147):HM$=CHR$(19)                              KI
•135 PRINT SC$;CO$(6)                                        OE
•140 POKE53280,2:POKE53281,2:PRINTSPC(249)"HOLD ON, JUST A SECOND"   KE
•150 DO$=HM$:FORX=1TO24:DO$=DO$+CHR$(17):NEXT                GH
•160 U3$="":FORX=1TO3:U3$=U3$+CHR$(145):NEXT                 HD
•170 R3$="":FORX=1TO3:R3$=R3$+CHR$(29):NEXT                  NF
•180 L5$="":FORX=1TO5:L5$=L5$+CHR$(157):NEXT                 GB
•190 SP$="":FORX=1TO39:SP$=SP$+" ":NEXT                      BP
•200 SZ$=CHR$(17)+CHR$(29)+CHR$(110)+CHR$(163)+CHR$(109)+CHR$(157)+CHR$(157)   OC
•210 SZ$=SZ$+CHR$(157)+CHR$(17)+CHR$(125)                    LM
•220 T$=CHR$(111)+CHR$(183)+CHR$(183)+CHR$(183)+CHR$(112)+L5$+CHR$(17)   OJ
•230 T$=T$+CHR$(165)+R3$+CHR$(167)+L5$+CHR$(17)              FC
•240 T$=T$+CHR$(165)+R3$+CHR$(167)+L5$+CHR$(17)              FC
•250 T$=T$+CHR$(108)+CHR$(175)+CHR$(175)+CHR$(175)+CHR$(186)+L5$+U3$   HB
•260 BS$=B2$:FORX=1TO5:BS$=BS$+CHR$(166):NEXT                EE
•270 BS$=BS$+L5$+CHR$(17):BT$="":FOR X=1TO4:BT$=BT$+BS$:NEXT:BS$=BT$+B1$   NG
•280 DIMCH$(24):FOR X=1TO24:READCH$:CH$(X)=CH$:NEXT          DP
•290 DATA A,B,C,D,E                                          HI
•295 DATA F,G,H,I,J                                          KD
•300 DATA K,L,M,N,O                                          NC
•305 DATA P,Q,R,S,T                                          OF
•310 DATA U,V,W,X                                            GE
•320 IF PEEK(49475)<>234 OR PEEK(49806)<>24THEN GOSUB9010    JE
•330 PX=0:POKE53280,PX:POKE53281,PX                          BC
•340 PRINTSC$:K$=""                                          OI
•350 PRINTDO$;SP$;B1$;:GOSUB8010:GOTO360                     OO
```

```
•360 PRINTHM$;                                   JH
•370 ROW=5:COL=1:BR=5:BC=5:MV=0:CU$=T$           AO
•380 PRINT DO$;SPC(7);CO$(2);"JOYSTICK OR
 KEYBOARD  (J/K)"C2$;                            DD
•390 GETKK$:IFKK$=""THEN390                      AO
•400 IFKK$="J" THEN420                           OD
•410 IFKK$<>"K" THEN390                          IF
•420 PRINTDO$;SP$;:GOSUB3040                     DA
•430 REM ***  MIX Y/N PRINT  ***                 MK
•440 PRINTCO$(4);DO$;LEFT$(SP$,15);"MIX
 (Y/N)";                                         HJ
•450 GETA$:IFA$=""THEN450                        HE
•460 IFA$= "Y"THEN PRINTDO$;SP$;:GOSUB304
0:GOSUB4010:MV=0:PRINTDO$;SP$;:GOTO490           JP
•470 IFA$= "N"THEN PRINTDO$;SP$;:GOTO490         KA
•480 GOTO450                                     CN
•490 CU$=T$:GOSUB3040:IFKK$="K" THEN7000         OA
•500 GOTO2010                                    FG
•1000 REM ***  SCROLL ROUTINE  ***               JF
•1010 MV=MV+1:PRINTDO$B2$"MOVE"B1$;MV;           JP
•1020 ON D GOTO1030,1050,1060,1080               OG
•1030 ML=49661:R=5:VU=0:COL=COL-HU               OG
•1040 GOTO1090:REM *LEFT                         HG
•1050 ML=49735:R=5:VU=0:GOTO1090:REM *RT         JM
•1060 ML=49416:R=4:HU=0:ROW=ROW-VU               GH
•1070 GOTO1090:REM *UP                           AP
•1080 ML=49483:R=4:HU=0:GOTO1090:REM *DN         KP
•1090 L=(COL-1)*5+8:T=(ROW-1)*4+2                DO
•1100 H=(HU+1)*5:V=(VU+1)*4                      LF
•1110 POKE251,L:POKE252,T:POKE253,H-1:POK
E254,V-1                                         HF
•1120 GOSUB6510                                  GD
•1130 FOR X=1TOR                                 LC
•1140 POKE254,V-1:POKE2,1                        NE
•1150 SYS ML                                     DM
•1160 NEXT                                       IA
•1170 POKE S+4,16                                FG
•1180 RETURN                                     IM
•1500 GET K$:IF K$=""THEN1500                    II
•1510 RETURN                                     IM
•2000 REM ***  READ JOY STICK  ***               BO
•2010 JV=PEEK(56320)                             GA
•2020 GETK$:IFK$=CHR$(147)THEN350                JP
•2030 FR=JVAND16:IF FR<>16THEN GOSUB5010:
GOTO2010                                         NB
•2040 JV=15-(JVAND15)                            GH
•2050 IF JV=0 THEN GOSUB2110:GOTO2010            KN
•2060 IF JV=1 AND ROW>1 THEN ROW=ROW-1:GO
SUB3010:GOTO2010                                 OP
•2070 IF JV=2 AND ROW<5 THEN ROW=ROW+1:GO
SUB3010:GOTO2010                                 EJ
•2080 IF JV=8 AND COL<5 THEN COL=COL+1:GO
SUB3010:GOTO2010                                 NH
•2090 IF JV=4 AND COL>1 THEN COL=COL-1:GO
SUB3010:GOTO2010                                 LI
•2100 GOTO2010:REM * INVALID DIRECTION *         CP
•2110 PRINTC1$;T$;                               CO
•2120 PRINTC2$;T$;                               DL
•2130 RETURN                                     IM
•3000 REM *******  SET CURSOR  *******           LE
•3010 IF FL THEN CU$=BS$                         PO
•3020 PRINTC1$;CU$;C2$;                          EJ
•3030 CU$=T$                                     PA
•3040 R=(ROW-1)*4+2:C=(COL-1)*5+8                MP
•3050 POKE251,R:POKE252,C                        JI
•3060 SYS 49806                                  KL
•3070 PRINTCU$;                                  IO
•3080 IF ROW=BR AND COL=BC THEN FL=1:GOTO
3100                                             BA
•3090 FL=0                                       HJ
•3100 RETURN                                     IM
•4000 REM *******  RANDOM MIX  *******           JJ
•4010 A=RND(RND(0)):PRINTLEFT$(DO$,5);B2$
;"MIXING"B1$;:GOSUB3040                          MH
•4020 FOR MX=1 TO 20                             CA
•4030 GOSUB4180                                  GC
•4040 IFA2=1THENROW=BR:COL=BC-A:GOTO4060         HJ
•4050 ROW=BR:COL=BC+A                            HN
•4060 IF COL<1 OR COL>5 THEN4030                 CL
•4070 GOSUB3010:GOSUB5010                        AH
•4080 GETK$:IFK$<>"" THEN4160                    FN
•4090 GOSUB4180                                  GC
•4100 IFA2=1THENCOL=BC:ROW=BR-A:GOTO4120         OA
•4110 COL=BC:ROW=BR+A                            KH
•4120 IF ROW<1 OR ROW>5 THEN4090                 IB
•4130 GOSUB3010:GOSUB5010                        AH
•4140 GETK$:IFK$<>"" THEN4160                    FN
•4150 NEXT                                       IA
•4160 PRINTLEFT$(DO$,5);LEFT$(SP$,6);:GOS
UB3040                                           FO
•4170 RETURN                                     IM
•4180 A=INT(RND(1)*4)+1                          AP
•4190 A2=INT(RND(1)*2)                           EB
•4200 RETURN                                     IM
•5000 REM ***  PREPARE TO SCROLL  ***            AE
•5010 IF COL<>BC THEN5060                        LO
•5020 IF BR=ROW THEN RETURN                      BI
•5030 VU=ABS(ROW-BR):D=4:TR=ROW+1                IK
•5040 IF BR<ROW THEN D=3:TR=ROW-1                BD
•5050 BR=ROW:GOSUB1010:ROW=TR:GOSUB3040:R
ETURN                                            AK
•5060 IF ROW<>BR THEN GOSUB6010:RETURN           FF
•5070 HU=ABS(COL-BC):D=2:TC=COL+1                CE
•5080 IF BC<COL THEN D=1:TC=COL-1                OD
•5090 BC=COL:GOSUB1010:COL=TC:GOSUB3040:R
ETURN                                            KI
•5100 IF COL<>BC THEN RETURN                     HF
•6000 REM *******  BUZZER  *******               ND
•6010 FORA=1TO8:POKE54296,15:FORT=1TO3:NE
XT:POKE54296,0:FORT=1TO3:NEXT:NEXT               LF
•6020 RETURN                                     IM
•6500 REM ***  PONG SUBROUTINE  ***              ML
•6510 S=54272                                    BI
•6520 POKE S+1,20                                FE
•6530 POKE S+5,9                                 DG
•6540 POKE S+6,9                                 DP
```

```
•6550 POKE S+24,4                                  FH
•6560 POKE S+4,17                                  FF
•6570 RETURN                                       IM
•7000 REM ***  GET KEYBOARD INPUT  ***             FB
•7010 GETK$:IFK$="" THEN GOSUB2110:GOTO70
 10                                                JD
•7020 IFK$=CHR$(147)THEN350                        JD
•7030 IFK$=" " THEN GOSUB5000:GOTO7010             OA
•7040 IF K$="A" AND ROW>1 THEN ROW=ROW-1:
 GOSUB3010:GOTO7010                                EE
•7050 IF K$="Z" AND ROW<5 THEN ROW=ROW+1:
 GOSUB3010:GOTO7010                                GC
•7060 IF K$="." AND COL<5 THEN COL=COL+1:
 GOSUB3010:GOTO7010                                FK
•7070 IF K$="," AND COL>1 THEN COL=COL-1:
 GOSUB3010:GOTO7010                                BJ
•7080 GOTO7010                                     FL
•8000 REM *******  DRAW BOARD  *******             IC
•8010 CH=1:PRINTB1$;:IFK$=CHR$(147)THEN81
 00                                                PJ
•8020 PRINTHM$;SPC(6);CHR$(117);:FORX=1TO
 27:PRINTCHR$(96);:NEXT:PRINTCHR$(105)             IG
•8030 PRINTSPC(6)CHR$(98);" ";:FORX=1TO25
 :PRINTCHR$(164);:NEXT:PRINT" "CHR$(98)            MC
•8040 PRINTLEFT$(DO$,2):FOR X=1 TO 20              LH
•8050 PRINT SPC(6);CHR$(98);CHR$(167);SPC
 (25);CHR$(180);CHR$(98)                           EF
•8060 NEXT                                         IA
•8070 PRINTHM$;LEFT$(DO$,23);                      BC
•8080 PRINTSPC(6)CHR$(98);" ";:FORX=1TO25
 :PRINTCHR$(163);:NEXT:PRINT" "CHR$(98)            MF
•8090 PRINTSPC(6);CHR$(106);:FORX=1TO27:P
 RINTCHR$(96);:NEXT:PRINTCHR$(107)                 HK
•8100 FOR ROW=1 TO 5                               DM
•8110 FOR COL=1 TO 5                               DA
•8120 IF ROW=5 AND COL=5 THEN PRINTB3$;:C
 U$=BS$:GOSUB3040:GOTO8170                         KM
•8130 PRINTB3$;:CU$=BS$:GOSUB 3040                 KJ
•8140 PRINTB1$;:CU$=T$:GOSUB3040                   BN
•8150 PRINTB2$;:CU$=SZ$:GOSUB3040                  DI
•8160 PRINTCO$(ROW);CH$(CH);B2$;CHR$(125)
 :CH=CH+1                                          LO
•8170 NEXT COL                                     GF
•8180 NEXT ROW                                     HN
•8190 RETURN                                       IM
•9000 REM *******  ML ROUTINES  *******            ON
•9010 FORI=49416TO49815                            GD
•9020 READT:POKEI,T:NEXT                           CF
•9030 RETURN                                       IM
•49416 DATA32,226,193,165,251,166,252,32,
 155,193,32,205,193,166,254,202,48                 EI
•49433 DATA21,24,165,208,105,40,133,210,1
 33,214,165,209,105,0,133,211,32                   JL
•49450 DATA119,193,202,16,235,164,253,185
 ,32,192,145,208,185,72,192,145,212                DA
•49467 DATA136,16,243,32,226,193,88,96,23
 4,234,234,234,234,234,234,234,32                  KC
•49484 DATA226,193,32,132,194,170,165,251
 ,32,155,193,32,205,193,166,254,202                PL
•49501 DATA240,208,56,165,208,233,40,133,
 210,133,214,165,209,233,0,133,211                 BL
•49518 DATA24,32,119,193,202,16,234,48,18
 4,41,3,9,216,133,215,164,253                      DM
•49535 DATA177,210,145,208,177,214,145,21
 2,136,16,245,165,211,133,209,165,210              GM
•49552 DATA133,208,165,215,133,213,165,21
 4,133,212,96,72,169,216,133,208,172               KC
•49569 DATA136,2,136,132,209,24,165,208,1
 05,40,133,208,165,209,105,0,133                   CF
•49586 DATA209,202,16,240,24,104,101,208,
 133,208,133,212,165,209,105,0,133                 PI
•49603 DATA209,41,3,9,216,133,213,164,253
 ,96,165,2,208,3,169,32,44                         NE
•49620 DATA177,208,153,32,192,177,212,153
 ,72,192,136,16,236,96,120,162,30                  HL
•49637 DATA181,208,72,189,0,192,149,208,1
 04,157,0,192,202,16,241,96,234                    FK
•49654 DATA234,234,234,234,234,234,234,32
 ,226,193,32,132,194,165,251,166,211               EO
•49671 DATA32,155,193,160,0,177,208,133,2
 10,177,212,72,144,13,200,177,208                  GG
•49688 DATA72,177,212,136,145,212,104,145
 ,208,200,196,253,144,239,165,2,208                NJ
•49705 DATA3,169,32,44,165,210,145,208,10
 4,145,212,198,211,198,254,16,201                  CO
•49722 DATA32,226,193,88,96,234,234,234,2
 34,234,234,234,234,32,226,193,32                  OI
•49739 DATA132,194,165,251,166,211,32,155
 ,193,177,208,133,210,177,212,72,164               PO
•49756 DATA253,240,15,136,177,208,72,177,
 212,200,145,212,104,145,208,136,208               BB
•49773 DATA241,165,2,208,3,169,32,44,165,
 210,145,208,104,145,212,198,211                   AN
•49790 DATA198,254,16,203,48,182,166,252,
 138,24,101,254,133,211,96,234,24                  DC
•49807 DATA164,252,166,251,32,240,255,96,
 234                                               LB
```

MOVING THROUGH *Color Memory*

Continued from page 80

BASIC does with memory where it thinks screen memory is. The VIC chip is still looking at 1024, which BASIC isn't changing.

Now, if you are using this technique in a long program, then of course you'll have to move video memory to a higher block. As it stands, your program would only have the space between the end of screen memory at 2048 and the beginning of BASIC's screen memory at 15360. That's fine for this example program, but a long game would need more elbow room. For the technique of moving video memory, see the column two months ago, which played games with memory like you wouldn't believe.

This technique will also work with the VIC, which also has its BASIC screen memory pointer at 648. However, it is almost worthless to try this technique unless you have expanded memory beyond 8K; if you do have an expanded VIC, you should have little trouble translating this program from the 64 version included in this issue.

WORMS IN COLOR MEMORY

You can do a lot more with color memory than just highlight options. Remember that the character patterns only decide which dots in a character cell display foreground color and which display background color. It's color memory that does the rest. For example, you could fill the entire screen with characters, but if the background color and the foreground color were the same, the screen would look completely blank. Only when you changed color memory would any portion of the screen become visible.

So here is a rudimentary game that lets you control a worm moving around the screen. Every character on the screen is the same – CH$(90), the diamond-shaped dot. However, most of them are invisible. Some of them are orange – these are the segments of the worm. Some of them are green – these are food. When the worm eats food, it gets longer.

The worm is run through the array variable WM*(n)*. Each element of the array represents a segment of the worm; each element contains the address of that segment in color memory. Every time the worm moves, all the addresses are changed in the array WM*(n)*, as segment 2 is now "located" where segment 1 used to be. But only the first and last segments need to be POKEd – the first segment is changed to the worm color, CW, and the last segment is changed to the background color, CB. When 12 bits of food have been "eaten" and the worm is 15 segments long, the game starts over.

Use f7 and f5 for vertical movement, SHIFT and COMMODORE for horizontal movement. Press RUN/STOP to end the game. Both VIC and 64 versions are included in the program listings section.

Next month we'll take a close look at a fantastic program from Electronic Arts: *Seven Cities of Gold*. And we'll look at ways to make a program respond to the player's behavior, so that the whole mood of a game can take on the personality the player shows while playing.

What we're talking about is, in a sense, behavioral conditioning. By setting up a reward and punishment system, we'll be encouraging some actions and discouraging others. It's just as the parents of videogame players always suspected – we game designers *are* out to turn our players into puppets!

Thanks to Walter Meyers and Jan Iverson for sending me their excellent games. Both of you set difficult problems for yourselves and solved them admirably. I only wish you were offering them for publication, so that other readers could see some of the dazzling things that can be done with resourceful and creative game programming.

Also, thanks to Derward F. McKinney, who kindly sent in all the changes that would be necessary to allow *The Emerald Elephant of Cipangu*, from one of these columns last fall, to run on the VIC 20 (with at least 16K). If anyone would care to have a copy of his letter, send a stamped, self-addressed envelope to:

Elephant for VIC
546 Lindley Road
Greensboro, NC 27410

I'll xerox Mr. McKinney's letter and send you a copy. Since I don't have memory expansion on my VIC, I can't test his changes, but I assume they work. ☐

SEE PROGRAM LISTINGS ON PAGE 88

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO. 65 MT. MORRIS, ILL.

POSTAGE WILL BE PAID BY ADDRESSEE

Ahoy!

P.O. BOX 341
MT. MORRIS, ILL. 61054

PLACE
STAMP
HERE

AHOY! MAGAZINE
P.O. Box 2501
Clinton, Iowa 52735

DON'T MISS THE BOAT!



SUBSCRIBE TO AHOY!

☐ **Send me twelve issues of AHOY! for $19.95 ($26.95 Canada & elsewhere). I understand that I will be billed later.**

☐ **Send me twenty-four issues of AHOY! for $37.95 ($49.95 Canada & elsewhere). I understand that I will be billed later.**

NAME ______________________

ADDRESS ______________________

CITY ______________________

STATE __________ **ZIP** __________

READER SERVICE CARD

To request additional information on any product advertised or otherwise pictured in this issue of *Ahoy!* that is accompanied by a reader service number, circle the corresponding number below and mail this card. We will promptly forward your request to the designated companies.

101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270
271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300

PLEASE ANSWER THE QUESTIONS BELOW.

Sex and age
A ☐ male   B ☐ female   age________

Educational level completed
C ☐ elementary   F ☐ college graduate
D ☐ high school   G ☐ masters degree
E ☐ junior college   H ☐ phd

From which of the following sources did you obtain your copy?
K ☐ newsstand   N ☐ subscription (mail)
L ☐ from a friend or family member
M ☐ other ______________________

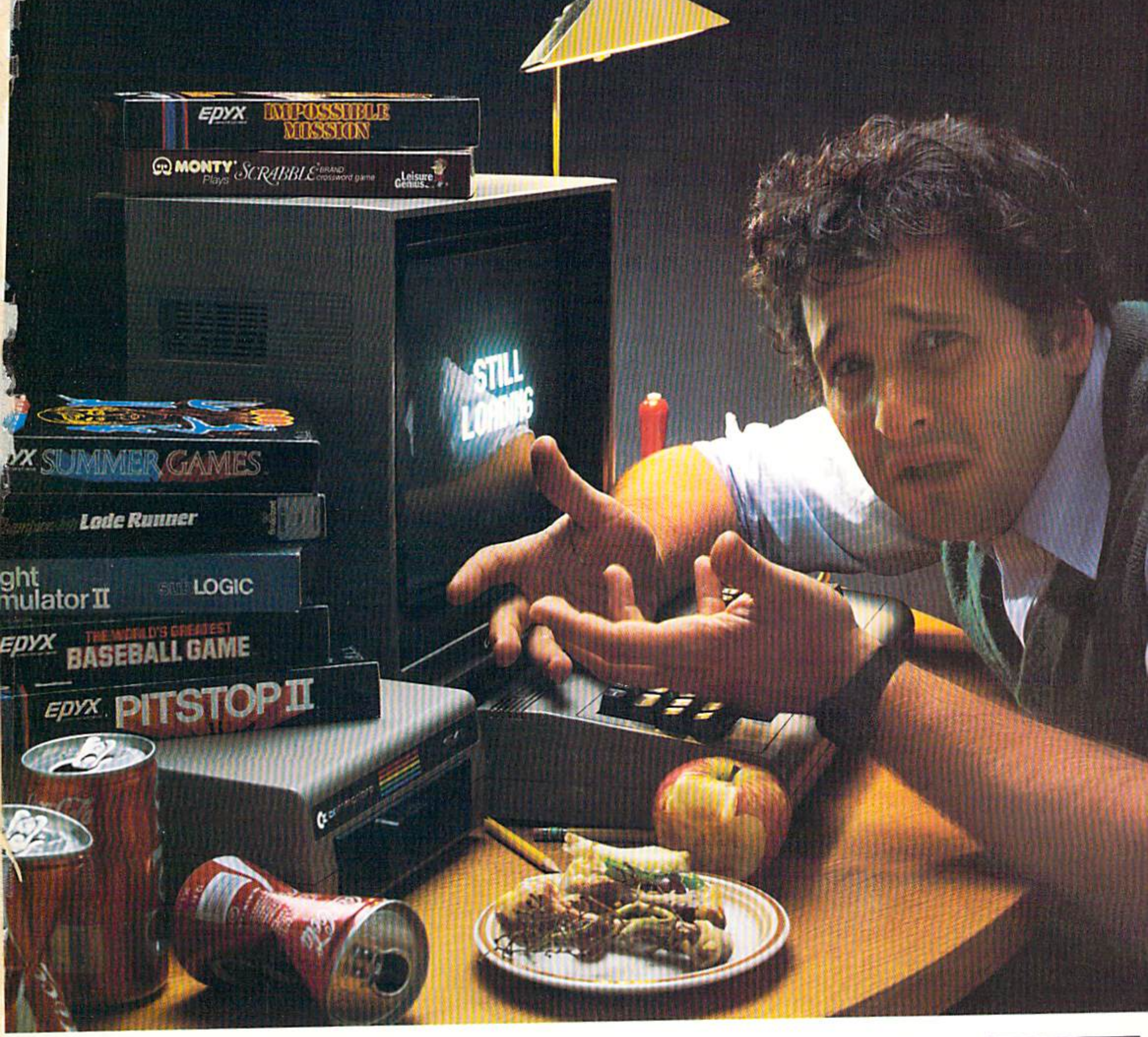
If not currently a subscriber, do you plan to become one?
I ☐ yes   J ☐ no

Name ______________________

Street ______________________

City __________ **State** __________ **Zip** __________

TIRED OF WAITING FOREVER FOR YOUR PROGRAMS TO LOAD?



INTRODUCING THE FAST LOAD CARTRIDGE FROM EPYX.

You're tired of waiting forever for your Commodore 64 programs to load. But it's no use glaring at your disk drive. Calling it names won't help, either. It was born slow – a lumbering hippo. You need the FAST LOAD CARTRIDGE from EPYX. FAST LOAD transforms your Commodore 64 disk drive from a lumbering hippo into a leaping gazelle. With FAST LOAD, programs that once took minutes to load are booted up in a matter of seconds.

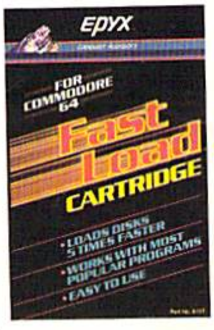
FAST LOAD can load, save and copy your disks five times faster than normal. It plugs into the cartridge port of your Commodore 64 and goes to work automatically, loading your disks with ease. And that's only the beginning. You can copy a single file, copy the whole disk, send disk commands, and even list directories without erasing programs stored in memory.

And unlike other products, the FAST LOAD CARTRIDGE works with most programs, even copy protected ones, including the most popular computer games.

The FAST LOAD CARTRIDGE from Epyx. Easy to insert, easy to use and five times faster. So why waste time waiting for your disks to load?

Speed them up with FAST LOAD!

Reader Service No. 209



EPYX